AD-A220 526

(1)

USING SIMULATION TO DETERMINE A STRATEGY FOR POSITIVELY TRACKING A CRUISE MISSILE BY CMMCA

THESIS

Antoine M. Garton
Captain, USAF

AFIT/GST/ENS/90M-6

S

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

90 04 13 205

AFIT/GST/ENS/90M-6

USING SIMULATION TO DETERMINE A STRATEGY FOR POSITIVELY TRACKING A CRUISE MISSILE BY CMMCA

THESIS

Antoine M. Garton
Captain, USAF

AFIT/GST/ENS/90M-6

Approved for public release; distribution unlimited

AFIT/GST/ENS/90M-6

USING SIMULATION TO DETERMINE A STRATEGY FOR POSITIVELY TRACKING A CRUISE MISSILE BY CMMCA

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Antoine M. Garton, B.S.
Captain, USAF

March 1990

Approved for public release; distribution unlimited

Preface

The study is a follow on to a thesis by Leonard G. Heavner who performed a preliminary investigation into the application of dynamic programming to determine the optimal controls to track a cruise missile by the Cruise Missile Mission Control Aircraft (CMMCA).

As suggested by Heavner an improved definition of a penalty function has been established. A different technique for obtaining the optimal controls is investigated to preclude the memory problems Heavner encountered using dynamic programming.

I am deeply indebted to my co-advisors Lt Col T. F. Schuppe and Lt Col W. P. Baker. Lt Col Schuppe gave me sage advice and always reminded me to step back and look at the big picture. Lt Col Baker unselfishly gave up countless hours to help me along this difficult problem. I would also like to thank Brian Jordan and Brian Kearns for making sure I went to the gym and worked out my frustrations there and not at home. Finally, I wish to thank my wife, Stacy, who made sure I always kept a sense of humor and did not take myself too seriously.

Antoine M. Garton

Table of Contents

List of Figures

List of Tables

Abstract

Air Launched Cruise Missiles (ALCM) are frequently tested in live firings over the western United States. These test flights require the use of several aircraft for launch, tracking, and control of the test vehicles. Tracking has historically been provided by a modified EC-135 aircraft. However, an EC-18 known as the Cruise Missile Mission Control Aircraft (CMMCA) has been developed and will be assuming the tracking mission.

The CMMCA will use a tracking radar rather than data provided by the telemetry antenna on the EC-135 to track and keep positive control of the cruise missile. The new radar has decreased azimuth limits compared to the EC-135. Because of a mission requirement to maintain constant radar coverage of ALCM, the problem of tracking has become much more difficult. A method has been developed to evaluate various options for maintaining the ALCM within the limits of the proposed radar. This method involves the use of computer simulation to model the movement of both vehicles. The modeling process is capable of employing various laws or procedures for tracking the cruise missile.

This paper describes the simulation developed for this evaluation as well as other insights gained in the evaluation process. This process should lead to a method for mission planning for CMMCA flight crews which should maximize the CMMCA's ability to provide continuous tracking coverage of the ALCM.

USING SIMULATION TO DETERMINE A STRATEGY FOR POSITIVELY TRACKING A CRUISE MISSILE BY CMMCA

I. Introduction

The 4950th Test Wing located at Wright-Patterson AFB will soon own and operate two specially configured aircraft called the Cruise Missile Mission Control Aircraft (CMMCA). The mission of the CMMCA will be to track and to keep positive control of a cruise missile during an operational test flight. The term positive control includes making sure the cruise missile maintains its scheduled flight path during the test as well as ensuring other traffic is not within collision range or on a collision course with the missile. At present, the aircraft used in the CMMCA role do not have a tracking radar at their disposal. What these aircraft do have is equipment that uses the CMMCA's telemetry antenna to indicate the heading and distance of the cruise missile from the airplane. With this data crews can chart the progress of the cruise missile and compare it with the flight planned course. This procedure is not real time and they cannot maintain positive control of the missile in any area. Because the crews do not have a radar

display of the cruise missile and surroundings, operational testing requires many other aircraft to participate. Four fighters fly close to the missile to ensure it is flying the correct flight path at all times. In addition, an air refueling tanker is required in order to provide the fighters with refueling support for this very long mission. Each tasked flying unit provides spares of each type of aircraft so the entire mission does not have to be aborted due to a broken primary aircraft. These aircraft are the minimum required to perform a test. Depending on the operational test, many more of the same type may be needed in addition to other special aircraft like the Airborne Warning and Control Aircraft System (AWACS). Understandably, the testing agency would like to decrease the number of aircraft involved for manpower and monetary reasons.

In October of 1990, flight testing will begin on a tracking radar system installed in two designated CMMCA's to aid in cruise missile flight tests. Even though the radar is a more powerful visual tool than the telemetry antenna and equipment, it is more limited in operating range and azimuth. With this radar however, the navigator will be able to visually track the cruise missile in real time. The radar can simultaneously show the position of the CMMCA, the cruise missile (CM) location and the CM planned flight path,

all on one screen. Because of the new radar system, the question has arisen whether the capability will exist to fly the mission with CMMCA only. A big advantage in this effort is that the entire scheduled flying route the cruise missile flies is known before an operational test. The CMMCA crews flight plan with the cruise missile flight path in mind. Tracking and keeping positive control are not difficult when the cruise missile is flying a straight path from one navigation point to the next. A problem occurs when the cruise missile makes a quick direction change. One of the most difficult maneuvers to track on radar is when the cruise missile makes a 270 degree turn, flies straight for two miles, and then does another 270 degree turn. The CMMCA cannot make these turns as quickly since the cruise missile and CMMCA have such different flight characteristics.

Another problem arises because the CMMCA flies at 28000 feet during the test and the cruise missile flies at altitudes less than a 1000 feet. Winds at 28000 feet can be quite different from those experienced at 1000 feet. To compensate for the change in winds the CMMCA must fly a different heading relative to the missile in order to follow the same flight path.

Research Problem

As mentioned above, the telemetry information provided for tracking is inadequate for use by the Test Wing. A

tracking radar can provide the test program a better capability to monitor test flights with fewer aircraft. With the acquisition of the new radar however comes a deficiency in the knowledge base on how to best use the radar. This particular radar has never been used in this type aircraft. The simulation model proposed will help decrease this lack of knowledge before flight testing begins.

One obvious solution to the tracking problem is to modify the test cruise missile with a homing device or beacon. This eliminates the problem of tracking it by radar and alleviates the radar azimuth and range limitations. Positive control as defined earlier is not assured however. Ground radar sites along the test path are also being evaluated so a beacon transmitting the cruise missiles position interferes with that portion of the test. Finally the Air Force does not want to modify these operational cruise missiles just because they are being tested.

Research Objective

The objective of this research is to develop an effective guidance algorithm for the CMMCA to track a cruise missile as it performs a series of worst case maneuvers in a flight test scenario. A simulation model of the two vehicles will be used to test and validate the algorithm.

Scope, Limitations and Assumptions

1. The CMMCA and the cruise missile each maintain a constant altitude during test flights.
2. A no wind condition will be assumed at the constant cruise missile altitude.
3. The simulation model will use reasonable values for aircraft flight parameters and characteristics.
4. The CMMCA can simultaneously turn and change velocity or do either individually to keep the cruise missile within the radar envelope.

Justification

At the moment, one cruise missile test flight requires many supporting aircraft. Deriving a method requiring only the CMMCA for cruise missile testing can save both time and money.

Overview

Chapter 2 presents background on possible solution techniques for solving the research objective. Control theory, simulation and dynamic programming will be discussed.

Chapter 3 expounds on the solution technique, a combination of simulation and control theory. The results of this research are presented in Chapter 4.

Finally, Chapter 5 discusses some conclusions and recommendations for future research in this tracking problem area.

II. Background

This chapter examines three possible approaches to the CMMCA tracking problem. Control theory, dynamic programming, and simulation were considered as alternative techniques.

Control Theory

Classical control system design is frequently a trial-and-error process using methods of analysis iteratively to produce 'tolerable' system parameters. Today with the advent of the digital computer, more complex systems can be analyzed using the newer approach of optimal control theory. Optimal control theory seeks to maximize or minimize the operation of a physical process. The final objective of optimal control theory is to determine the input signals that will cause a process to minimize or maximize some performance measure while staying within certain physical constraints of the system (6:3). Kirk lists the following steps in formulating an optimal control problem (6:4):

1. A mathematical description (or model) of the process to be controlled.
2. A statement of the physical constraints.
3. Specification of a performance criterion.

A mathematical description of the process to be controlled is no easy task. The first thing that must be identified are the variables necessary to define the system

and describe the motion, if it is a dynamic system. In control theory these variables are called state variables. The CMMCA problem has many variables associated with it including true airspeed, position, heading, and bank angle. Choosing which of these variables best describes the system being modelled is an art in control theory. Someone else with a different purpose or objective may model the same system with quite different state variables. Kirk recommends finding the simplest mathematical description that satisfies the physical system being modelled for the range of possible inputs (6:4). For example, the flight of the CMMCA can be modelled by choosing true airspeed and bank angle as control inputs and heading as a state variable. Since flight follows well known physical laws, state equations using the previously mentioned variables can be derived to describe or model the CMMCA motion. The state equations in ordinary differential form are:

$$\frac{dx}{dt} = TAS \sin[HDG] + WV_x \tag{2.1}$$

$$\frac{dy}{dt} = TAS \cos[HDG] + WV_y \tag{2.2}$$

$$\frac{dHdg}{dt} = \frac{g \tan[\alpha]}{TAS} \tag{2.3}$$

The first equation describes the velocity of the CMMCA in the x direction (East) given the control input of true airspeed (TAS) and the state variable heading (HDG). The last term in Eq (2.1) is the wind velocity component in the x direction. The second equation does the same except in the y direction (North). The third differential equation defines the rate of change of the CMMCA heading given the two control inputs, bank angle (α) and true airspeed.

After the model has been selected, physical constraints on the system must be addressed. One obvious constraint from the example above is the operating airspeed limitations for the CMMCA. Depending on the aircraft gross weight, there is always an upper and lower limit on flying speeds. Another constraint is the maximum bank angle the CMMCA can fly. Once all the physical constraints have been identified and included in the mathematical model the next step is to select a performance criterion.

Kirk defines optimal control as "one that minimizes (or maximizes) the performance measure" (6:10). In the CMMCA problem a rather obvious performance measure is minimizing the total amount of time during a flight maneuver the cruise missile is not within the CMMCA radar cone. Heavner expressed the performance measure as a penalty function with range and azimuth as inputs. Using this penalty function,

dynamic programming tried to minimize the time weighted average deviations from a given nominal range and azimuth.

The optimal control problem needs to identify admissible controls in the form of bank and airspeed that will allow the CMMCA to follow an acceptable flight path that minimizes some performance measure.

In an unpublished report, Baker suggests using control theory to provide an optimal tracking algorithm. It is a continuous approach requiring an objective function and associated constraints. The entire flight path for the CMMCA given this objective function could be optimized using a mathematical technique called calculus of variations (11:11).

Optimal control theory seems ideal for the cruise missile tracking problem and a variation of Baker's technique was used by Heavner in his dynamic programming approach.

Dynamic Programming

Dynamic programming uses mathematical techniques to solve interrelated decisions (4:266). In control theory dynamic programming is used to maximize or minimize some control function of the state variables. Hillier and Lieberman describe the basic features which characterize dynamic programming:

1. The problem can be divided into stages, with a policy decision required at each stage.
2. Each stage has a number of states associated with it.
3. The effect of the policy decision at each stage is to transform the current state into a state associated with the next stage (possibly according to a probability distribution).
4. Given the current state, an optimal policy for the remaining stages is independent of the policy adopted in previous stages.
5. The solution procedure begins by finding the optimal policy for each state of the last stage.
6. A recursive relationship that identifies the optimal policy for each state as stage n, given the optimal policy for each state at stage (n + 1), is available.
7. Using this recursive relationship, the solution procedure moves backward stage by stage - each time finding the optimal policy for each state of that stage - until it finds the optimal policy when starting at the initial stage. (4:270)

Heavner defined the state of the system as the position and velocity of the CMMCA. The controls that determined the state were bank angle and thrust. The equations of motion of flight were used to determine the state of the system given these two controls. The stages in this dynamic programming problem were represented by time. At each time increment a decision had to be made as to what controls should be applied to determine the state of the system. Heavner's performance criterion is a penalty function to be minimized at each stage. The objective was to minimize the time-weighted deviations from a desired location ahead of the CMMCA. This desired location would be where the cruise missile should be in the radar sweep. The two deviations

from desired location were defined as range in nautical miles (nm) and azimuth in radians, both measured from the CMMCA position.

The problem Heavner encountered was the high dimensionality of the tracking problem requiring excessive amounts of high speed memory to compute an optimal solution. To alleviate the curse of dimensionality problem, Heavner used a technique called state increment dynamic programming. The procedure is fundamentally the same as regular dynamic programming with the addition of two new concepts: the block and the time over which a control is applied. Problems arose on how to implement the CMMCA problem using state increment dynamic programming. A more successful application of conventional dynamic programming was used by decreasing the state space and only looking at single turns of the cruise missile during the simulation. The model was written in FORTRAN code and run on a VAX under the VMS operating system. Heavner recommends the model only be used for insight into the controls that might be necessary for the CMMCA to use during a cruise missile maneuver (3:1-38).

Simulation

Another possible solution technique is continuous simulation. The technique was initially dismissed in Heavner's thesis because dynamic programming could theoretically achieve an optimal solution which continuous

simulation could not guarantee (3:36-37). Due to the dilemma in the dynamic programming solution technique, continuous simulation is being reconsidered.

Is continuous simulation a suitable technique for this particular problem? An article by Bekey points out that the "purpose of simulation is either to yield insight into the behavior of the process being simulated or to make predictions of performance" (2:57). Both results would be of interest in the CMMCA problem. Bekey goes on to say that a good measure of fit is if the simulation comes close to the actual behavior of the process over some time period (2:58). Since the equations of motion of flight follow well known physical laws, it makes one confident that the required flying portion of the CMMCA and cruise missile simulation would closely represent the real world. Simulation may not attain an optimal solution like the dynamic programming technique tried earlier, but it could be used to identify rules of thumb to follow in better tracking a cruise missile during a flight test. Many systems that are too hard to optimize are at least better understood using simulation.

Tasks Necessary for Simulation. Pritsker says that two tasks necessary for simulation are the collection of equations and events that describe the particular process being modeled and the evaluation of these equations and

events for different control inputs (9:484). A recent article by Sheppard suggests more detailed steps in simulation (12:14):

1. System definition (boundaries, restrictions, etc.)
2. Model formulation
3. Data preparation
4. Model translation to a machine executable form
5. Model validation
6. Design of experiments
7. Planning of computer runs
8. Model experimentation
9. Interpretation of model outputs
10. Implementation of model results
11. Documentation

All these processes are necessary to accomplish the task of solving the CMMCA problem. The first two simulation stages, system definition and model formulation, help in deciding what equations to use in defining the simulation model.

Equations. As mentioned earlier, the equations of motion of flight are well-defined physical laws. An article by Stewart shows a rather comprehensive and all encompassing derivation for the equations of flight. These equations include control input forces (rudder, elevator, aileron) as well as the aerodynamic forces (propulsion, drag, lift etc.) and gravitational forces (13:237). The CMMCA problem does not require some of these equations or certain variables. For instance, the CMMCA will not be climbing or descending while tracking the cruise missile so any motion along the

'altitude' axis will be zero. The cruise missile will be flying low and at a reasonably constant altitude for tracking, so its vertical axis motion will be assumed to be zero also. Turns and changes in velocity are the only two maneuvers allowed when tracking the cruise missile. Heavner has an appendix of the needed equations well defined and derived with the specific restrictions mentioned previously incorporated (3:42).

The next big consideration is how to verify and validate the answers the simulation is producing.

Validation and Verification. When simulation is used for problem-solving, the modeler is concerned whether the information or results taken from the model can be used with confidence. Hopefully model validation and verification can address this concern.

Sargent makes a clear distinction between validation and verification in the following simplified version of the modelling process and as shown in Figure 1.

> Conceptual model validity is defined as determining that the theories and assumptions underlying the conceptual model are correct and that the model representation of the problem entity is "reasonable" for the intended use of the model. Computerized model verification is defined as ensuring that the computer programming and implementation of the conceptual model is correct. Operational validity is defined as determining that the model's output behavior has sufficient accuracy for its intended purpose or use over the domain of the model's intended application (10:33).

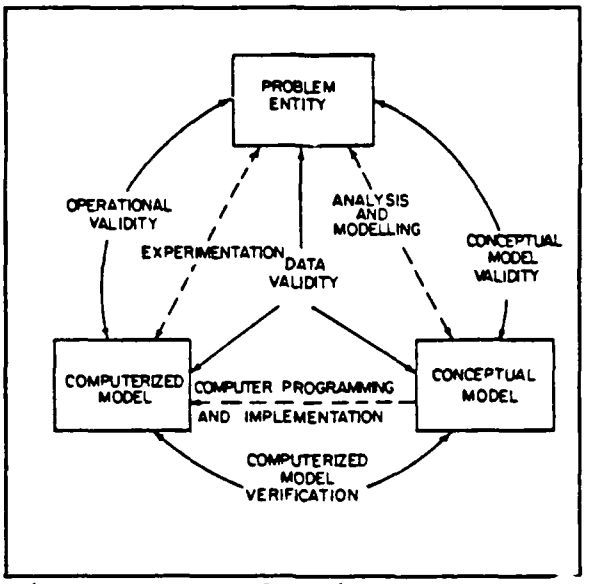

Figure 1. Modelling Process
(Reprinted from 8:34)

Sargent points out that no matter how many validation techniques are used there is still no guarantee that the model is absolutely valid over the entire range of possible application. Some of the validation techniques discussed by Sargent and deemed useful for this particular thesis effort are:

1. Operational Graphics - Model's operational behavior is graphed through time.
2. Comparison to other models - Heavner's model is available as a comparison.
3. Face validity - People knowledgeable about the system are asked if the output is reasonable.
4. Parameter variability - Values of parameters in the model are varied to determine any effect on the output (10:33-34).

Implementation. Schuppe has written a rudimentary simulation that takes all these considerations into account and tries to establish how well a CMMCA can track a cruise

missile (11:1). Both the CMMCA and cruise missile were easily modelled using the equations of motion of flight. Identification of a guidance algorithm the CMMCA could use during the simulation to track the missile was a more difficult problem. The simulation showed that just reacting to missile maneuvers was not adequate enough to effectively track the missile. Schuppe concluded that a guidance algorithm needed to take into account the future position of the missile so the CMMCA could alter its flight path before the missile.

Conclusion

The review brought into focus some of the more challenging aspects of tracking a cruise missile. It seems an optimal guidance algorithm of some kind is needed to determine what path the CMMCA should fly to best track the cruise missile. Continuous simulation can be used to show this flight path as well as evaluate the guidance algorithm for effectiveness during different maneuvers. Heavner's dynamic programming model can aid in validating any future model developed. This research effort will explore the use of a modified version of all three techniques to help solve the problem.

III. Methodology

The methodology for obtaining an optimal tracking algorithm took form by following steps recommended by Baker:

1. Identify a quantifiable objective functional and associated constraints.
2. Treat problem in discrete time increments.
3. Optimize over the entire flight maneuver with respect to chosen objective function.
4. Optimize using a modified gradient search procedure.

The term functional used here and later on denotes a function containing several functions. These functions and what they represent will be discussed in the next section.

Baker had already identified an objective functional that would take into consideration the future location of the cruise missile as well as performance constraints for the CMMCA (1:4). Discretizing the time increments was necessary to simplify the objective functional so it could be solved with known numerical techniques. Each cruise missile maneuver is optimized separately to avoid problems with excessive need for computer memory. This follows closely how a flight crew would approach flight planning a mission with many cruise missile maneuvers. A modified gradient search method was required to solve for the CMMCA control inputs, bank and airspeed.

Baker's Objective Functional

The objective functional is really a cost functional that penalizes the CMMCA for current position error and improper control inputs. The only control inputs allowed in the objective functional are CMMCA airspeed and bank angle. Only these two controls can be used to minimize how far away the CMMCA is from some desired position from the CM.

Current Position Error. θ_o and r_o are the desired nominal bearing and range to the missile. The nominal range can vary from the closest distance the radar can view the target to the maximum distance. The nominal bearing can range from a $-\theta_c$ degrees measured counterclockwise from the nose of the aircraft to a positive θ_c degrees measured clockwise from the nose of the aircraft. An r_o of eight nm's and a θ_o of zero degrees were selected to reflect where a navigator would most likely try to keep the target while tracking it.

$$\begin{aligned} r(t) &\in [r_1, r_2] \\ \theta(t) &\in [-\theta_c, \theta_c] \\ E(t) &= (r(t)-r_o)^2+(\theta(t)-\theta_o)^2 \end{aligned} \tag{3.1}$$

E(t) is the total error due to current CMMCA position. The current position error is further defined by the equation below where W_1 is a weight to be determined later during experimentation.

$$P(r,\theta) = [r(t) - r_o]^2 + W_1[\theta(t) - \theta_o]^2 \qquad (3.2)$$

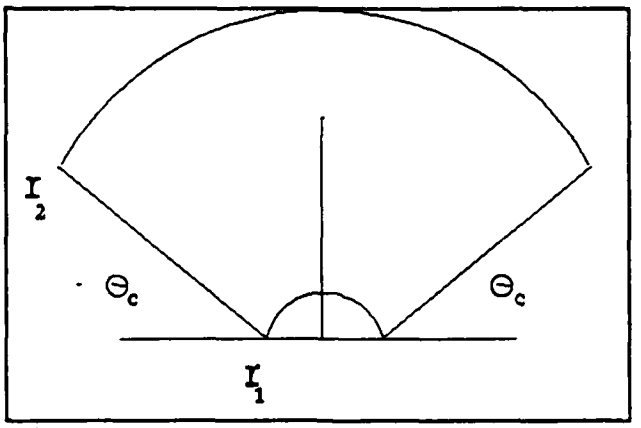

Figure 2. Radar Cone

In an x-y axis system, where y denotes north and x denotes east, range and azimuth from CMMCA to CM can be defined as:

$$r(t) = [(x_m(t) - x(t))^2 + (y_m(t) - y(t))^2]^{1/2}$$
$$\theta(t) = \tan^{-1}\left[\frac{x_m(t) - x(t)}{y_m(t) - y(t)}\right] - Hdg(t) \qquad (3.3)$$

The x and y positions subscripted with m represent cruise missile coordinates while the x and y variables with no subscript represent CMMCA position. This study will not take into account the slant range accrued from the CMMCA and cruise missile flying at different altitudes.

Control Input Penalties. Both airspeed and bank angle have penalties associated with them in the cost functional. The penalty function for airspeed is expressed as follows:

$$P_s = W_2 \left[\frac{V(t) - U_0}{\mu} \right]^{2K_3} \qquad (3.4)$$

Let U_o be the desired nominal true airspeed and mu an allowable variance from the nominal. A desirable nominal TAS would be a value somewhere in the middle of the CMMCA performance envelope. Let V(t) represent the true airspeed velocity at some time t. The highest and lowest true airspeed the aircraft can fly at any time in a mission is dependent on the airplanes gross weight at that time. This study will assume a low TAS of 330 knots and a high of 480 knots for the CMMCA during a maneuver. A desirable nominal TAS using these highs and lows is 400 knots, allowing ample range for speed increases and decreases. The variable K_3 represents an integer affecting the steepness of P_s for variations greater than mu as well as the flatness of the parabola inside the mu boundaries. Mu is defined as half the interval between high and low true airspeeds, or 75 knots. As demonstrated by Figure 3, the penalty for being relatively close to the nominal airspeed is small versus the penalty incurred for being past $U_o + \mu$ where the curve is very steep.

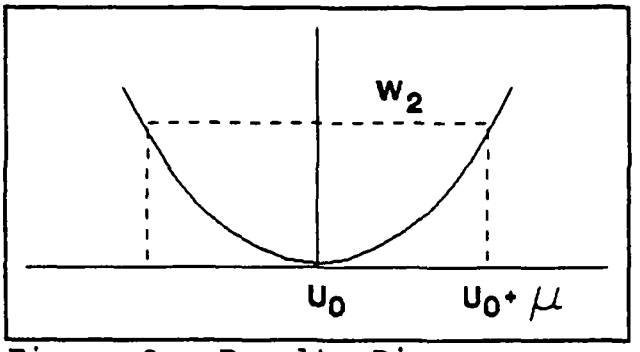

Figure 3. Penalty Diagram

The bank angle penalty function behaves the same way and is defined as:

$$P_{\alpha} = W_3 \left[\frac{\alpha(t)}{\beta_0} \right]^{2K_4} \tag{3.5}$$

Beta zero is defined as bank angle variance while K_4 is an integer influencing the function like the K_3 constant in Eq (3.4).

The total objective functional throughout the maneuver will be given by:

$$J = \int_{t_o}^{t_f} [P(r(t),\theta(t)) + P_{\theta}(TAS(t)) + P_{\alpha}(\alpha(t))]\, dt \tag{3.6}$$

The time interval the integral is evaluated over is defined as the time a cruise missile maneuver starts, usually zero, up to the time the maneuver is completed. Due to the extreme complexity of the objective functional, it would be

difficult or impossible to solve explicitly. Instead of working with a continuous time functional, Eq (3.6) was discretized so it could be solved using numerical analysis on a computer system. The discretized version of Eq (3.6) follows.

$$J[V,\alpha] = \sum_{i=0}^{N} Wt_i J_i$$

$$J_i = [r(t_i) - r_o]^2 + W_1(\theta(t_i) - \theta_0)^2 + W_2\left(\frac{V_i - U_0}{VA}\right)^{2K_5} + W_3\left(\frac{\alpha_i}{\beta_0}\right)^{2K_6} \qquad (3.7)$$

This equation is defined for discrete time intervals t_o, t_1, t_2,... $t_n = t_f$. At each of these time intervals $V(t_i) = V_i$, the velocity (nm/min) and $\alpha(t_i) = \alpha_i$, the bank angle (radians). Wt_i represents some quadrature integration weight which can be changed to better approximate Eq (3.7). A trapezoidal rule will be used for the numerical analysis approximation of the integral (7:146). The velocities and bank angles are contained in vector form with each vector as long as there are time increments in the maneuver. If the maneuver takes 80 time units, there are 80 velocities, 80 bank angles and 80 weights that define J the objective functional. The next step involves taking the partial of the objective functional J in order to develop a method for minimizing J.

Gradient of J

Recall the discretized version of the objective functional presented in Eq (3.7). This equation along with a few others defined below will establish the equations necessary to derive the gradient of J.

Range r_i has been defined as the difference in distance from the cruise missile to the CMMCA. The missiles position, which is a given, is represented by the m subscript. Pythagorean's theorem derives the magnitude for the range.

$$r_i = ([X_{m_i} - X_i]^2 + [Y_{m_i} - Y_i]^2)^{1/2} \tag{3.8}$$

Theta is defined as shown representing the angle measured left (negative) or right (positive) of the CMMCA heading. Theta can range from -180 degrees to +180 degrees.

$$\theta_i = \tan^{-1}\left[\frac{X_{m_i} - X_i}{Y_{m_i} - Y_i}\right] - Hdg_i \tag{3.9}$$

The next three equations were derived in appendix A and are needed here to derive the gradient J. Each equation

exhibits how the left hand side changes with initial condition, subscripted by a 0, and elapsed time.

$$X_i - X_0 + \sum_{I=0}^{i} V_I \sin(Hdg_I)\, W_I^{(I)} \Delta t + WV_x (t_i - t_0) \qquad (3.10)$$

$$y_i - y_0 + \sum_{I=0}^{i} V_I \cos[Hdg_I]\, W_I^{(I)} \Delta t + WV_y (t_i - t_0) \qquad (3.11)$$

$$Hdg_I - Hdg_0 + \sum_{s=0}^{I} g \frac{\tan[\alpha_s]}{V_s} W_s^{(I)} \Delta t \qquad (3.12)$$

Since the objective functional is a function of both velocity and bank angle at some time, the gradient of J consists of two parts. The first part is the partial of J with respect to each velocity k. The second part is the partial of J with respect to each bank angle k. Eqs (3.13) through (3.20) are the results of taking the partial of J with respect to a velocity k of Eqs (3.7) through (3.12).

Eqs (3.13) and (3.14) are the partial of the first portion of Eq (3.7) with respect to a velocity k. Each component of the gradient of J is set to zero in order to solve and find a minimum for the objective functional equation.

$$\frac{\partial J}{\partial V_k} = 0 \qquad k = 1, \ldots N \qquad (3.13)$$

$$\sum_{i=0}^{N} W_{c_i} \frac{\partial J_i}{\partial V_k} = 0 \qquad (3.14)$$

The partial of J with respect to a velocity k of the second part of Eq (3.7) is derived below.

$$\frac{\partial J_i}{\partial V_k} = 2(r_i - r_0)\frac{\partial r_i}{\partial V_k} + 2W_1(\theta_i - \theta_0)\frac{\partial \theta_i}{\partial V_k} + W_2 \frac{2K_3}{\mu}\left(\frac{V_i - U_0}{\mu}\right)^{2K_3 - 1} \qquad (3.15)$$

Eq (3.15) shows the need for the partial of r with respect to V_k and the partial of θ_i with respect to V_k. Both are derived from Eqs (3.8) and (3.9).

$$\frac{\partial r_i}{\partial V_k} = \frac{1}{r_i}\left[(x_{m_i} - x_i)\left(-\frac{\partial x_i}{\partial V_k}\right) + (y_{m_i} - y_i)\left(-\frac{\partial y_i}{\partial V_k}\right)\right] \qquad (3.16)$$

$$\frac{\partial \theta_i}{\partial V_k} = \left[\frac{(y_{m_i} - y_i)\left(-\frac{\partial x_i}{\partial V_k}\right) - (x_{m_i} - x_i)\left(-\frac{\partial y_i}{\partial V_k}\right)}{(x_{m_i} - x_i)^2 + (y_{m_i} - y_i)^2}\right] - \frac{\partial H dg_i}{\partial V_k} \qquad (3.17)$$

The last three equations needed to complete taking the partial of J with respect to V_k are the partials of Eqs (3.10) through (3.12). The results are Eqs (3.18), (3.19), and (3.20).

$$\frac{\partial x_i}{\partial V_k} = \begin{cases} 0, k > i \\ \sin(Hdg_k) W_k^{(1)} \Delta t + \sum_{1=0}^{i} V_1 \cos(Hdg_1) W_1^{(1)} \Delta t \frac{\partial Hdg_1}{\partial V_k}, 0 < k \leq i \end{cases} \quad (3.18)$$

$$\frac{\partial y_i}{\partial V_k} = \begin{cases} 0, k > i \\ \cos(Hdg_k) W_k^{(1)} \Delta t + \sum_{1=0}^{i} -V_1 \sin(Hdg_1) W_1^{(1)} \Delta t \frac{\partial Hdg_1}{\partial V_k}, 0 < k \leq i \end{cases} \quad (3.19)$$

$$\frac{\partial Hdg_1}{\partial V_k} = \begin{cases} 0, k > 1 \\ 0, 1 = 0 \\ g \tan(\alpha_k) W_k^{(1)} \Delta t \left(\frac{-1}{V_k^2}\right), 0 < k \leq 1 \end{cases} \quad (3.20)$$

The same process as above must be accomplished for the partial of J with respect to some alpha (bank angle) of k. Eqs (3.21) through (3.28) are the results of these derivations.

$$\frac{\partial J}{\partial \alpha_k} = 0 \quad (3.21)$$

$$\sum_{1=0}^{N} W_{t_i} \frac{\partial J_i}{\partial \alpha_k} = 0 \quad (3.22)$$

$$\frac{\partial J_i}{\partial \alpha_k} = 2(r_i - r_0)\frac{\partial r_i}{\partial \alpha_k} + 2W_1(\theta_i - \theta_0)\frac{\partial \theta_i}{\partial \alpha_k} + W_3\frac{2K_4}{\beta_0}\left(\frac{\alpha_i}{\beta_0}\right)^{2K_4-1}\delta_{ik} \qquad (3.23)$$

$$\frac{\partial r_i}{\partial \alpha_k} = \frac{1}{r_i}\left[(x_{m_i} - x_i)\left(-\frac{\partial x_i}{\partial \alpha_k}\right) + (y_{m_i} - y_i)\left(-\frac{\partial y_i}{\partial \alpha_k}\right)\right] \qquad (3.24)$$

$$\frac{\partial \theta_i}{\partial \alpha_k} = -\left[\frac{(y_{m_i} - y_i)\left(-\frac{\partial x_i}{\partial \alpha_k}\right) - (x_{m_i} - x_i)\left(-\frac{\partial y_i}{\partial \alpha_k}\right)}{(x_{m_i} - x_i)^2 + (y_{m_i} - y_i)^2}\right] - \frac{\partial Hdg_i}{\partial \alpha_k} \qquad (3.25)$$

$$\frac{\partial x_i}{\partial \alpha_k} = \sum_{1=0}^{i} V_1 \cos(Hdg_1) W_1^{(i)} \Delta t \frac{\partial Hdg_1}{\partial \alpha_k} \qquad (3.26)$$

$$\frac{\partial y_i}{\partial \alpha_k} = \sum_{1=0}^{i} -V_1 \sin(Hdg_1) W_1^{(i)} \Delta t \frac{\partial Hdg_1}{\partial \alpha_k} \qquad (3.27)$$

$$\frac{\partial Hdg_1}{\partial \alpha_k} = \begin{pmatrix} 0, k > 1 \\ 0, 1 = 0 \\ g \sec^2(\alpha_k) W_k^{(1)} \Delta t \left(\frac{1}{V_k}\right), 0 < k \leq 1 \end{pmatrix} \qquad (3.28)$$

Modified Gradient Search

The second partials of J with respect to velocity and bank angle is deemed too complicated due to the size and complexity of the first partial derivative. Because of this complexity a modified gradient search method is needed in order to solve for the minimum of the objective functional without using the second partial derivative.

The gradient of J is determined by taking the partial derivative of J with respect to each velocity and bank angle in the maneuver vector. This has been accomplished in general form in the previous section. The maneuver vector will be defined to consist of all the velocities and bank angles for each time step in a particular maneuver. The gradient vector points in the direction of maximum increase for a function. Lambda provides the step size in the direction opposite the gradient vector. The old maneuver vector is replaced by a new maneuver vector according to the following calculation.

$$\begin{bmatrix} \underline{V} \\ \underline{\alpha} \end{bmatrix} = \begin{bmatrix} \underline{V_o} \\ \underline{\alpha_o} \end{bmatrix} - \lambda \left[\underline{\nabla J} \right] \qquad (3.29)$$

The gradient J is a vector containing the search direction for each velocity and each bank angle in the maneuver vector. This gradient vector multiplied by some lambda and

subtracted from the old maneuver vector provides an improved maneuver vector which produces a smaller J objective functional value than the previous maneuver vector.

Since this is an iterative process a stopping criteria needs to be implemented. At optimality the gradient J will vanish. The step size lambda will also approach zero at this point. Continuous checking of lambda and the gradient J within some tolerable value close to zero during program execution will provide the information necessary to decide if optimality has occurred and terminate the program.

Because the second derivative was not available, a technique needed to be established that will provide a step size lambda for the search direction. It was first decided to pick three lambda values; compute three separate new maneuver vectors using each; compute the functional J from these maneuver vectors; fit a parabola through the three points; identify the lowest point on the curve and use it as the best step size for that direction. Lambda optimal, as it was referred to, was the step size used to compute a new maneuver vector using Eq (3.29). One of the three lambda's took on the value of zero to save on computation time while the other two were picked based on results of experimentation. The main reason for using this approach was to avoid correcting in some direction using too small or too large a step size. The parabola fitting can save

iterations by calculating a better step size from the lambda's given and proceed more judiciously in the direction computed. Appendix C contains the derivations necessary to fit the parabola to the three objective functional points and compute an optimal lambda.

Implementation

A FORTRAN computer code was built to optimize the objective functional discussed above using the modified gradient search technique. The gradient search procedure requires an initial maneuver vector to start the search. In earlier studies of the CMMCA problem a basic SLAM II simulation model was built to replicate the flight characteristics of both the cruise missile and CMMCA. This model provided the initial guess for a maneuver vector for the FORTRAN optimization algorithm by outputting the banks and speeds the cruise missile flies during a maneuver. The initial guess assumes the CMMCA will fly the same flight path as the cruise missile. The simulation was also available to measure, in percentage of time, the performance of the guidance algorithm in keeping the cruise missile within the radar envelope once an optimal maneuver vector had been calculated. The FORTRAN code for the modified gradient search and the SLAM II simulation code are both contained in Appendix B.

Algorithm Example

As mentioned above, the algorithm requires an initial guess to start a search for a better maneuver vector. Along with the initial guess the algorithm also requires the cruise missile x and y position for the entire maneuver. Table 1 shows the required entries. The bank is entered in radians and the speed in nm/min. These two columns represent the initial guess of bank and speed the CMMCA should fly to follow the CM. The first row represents the initial conditions for the CMMCA and CM. Starting with zero each row represents .1 minutes of elapsed time. This small maneuver example lasts 1.2 minutes. The columns marked XCM and YCM represent the CM x and y positions in nautical miles for each time period. The CMMCA starting position is currently hard wired in the program to be x=0, y=0. In this case the CMMCA is in a twenty degree bank starting out 8 nm behind the CM at a true airspeed of 6.667 nm/min. The cruise missile is also going the same speed although flying straight and level as inferred by the position columns.

Intermediate results as well as final results were printed to show the sequence of events leading to a more optimal bank and speed profile. Table 2 shows calculations for one iteration while Table 3 shows the results achieved after one iteration.

Once the algorithm has the data it builds the maneuver vector which in this example will consist of twelve speeds and twelve banks. The initial conditions (i.e. initial velocity and bank angle) cannot be altered so it will not be included in the maneuver vector.

Table 1. Input Example

BANK	SPEED	XCM	YCM
0.00	6.667	0.0	8.000
0.35	6.667	0.0	8.667
0.35	6.667	0.0	9.333
0.35	6.667	0.0	10.000
0.35	6.667	0.0	10.667
0.35	6.667	0.0	11.333
0.35	6.667	0.0	12.000
0.35	6.667	0.0	12.667
0.35	6.667	0.0	13.333
0.35	6.667	0.0	14.000
0.35	6.667	0.0	14.667
0.35	6.667	0.0	15.333
0.35	6.667	0.0	16.000

The maneuver vector is one column consisting of 24 numbers, the first 12 are the speeds, the second 12 the banks. The algorithm proceeds to calculate a gradient direction for each bank and speed in the maneuver vector as shown in the Table 2 section called DELTJ. The first number represents the direction to step in for the first velocity

and so on until the 24th number which represents the direction to step in for the last bank angle.

Table 2. Calculations After One Iteration

DELTJ	JSTOP	GLAMB	LAMBDA OPT
-2.038080	59.751010	30.375850	.08333333
-1.903263		27.860455	
-1.747402		25.345060	
-1.569421			
-1.367907			
-1.142408			
-.08958371			
-.6375485			
-.3857875			
-.1695291			
-.0288455			
.009263428			
25.919670			
24.997360			
23.67988			
21.973090			
19.895210			
17.484740			
14.810160			
11.979380			
9.146695			
6.514660			
4.328537			
1.689987			

Table 3. Results After One Iteration

TIME	BANK	SPEED	RANGE	THETA
.1	.35	6.7	8.0	-2.8
.2	.35	6.7	8.0	-8.5
.3	.35	6.7	8.0	-14.8
.4	.35	6.7	8.1	-21.5
.5	.35	6.7	8.1	-28.6
.6	.35	6.7	8.2	-36.2
.7	.35	6.7	8.4	-44.1
.8	.35	6.7	8.6	-52.3
.9	.35	6.7	8.9	-60.6
1.0	.35	6.7	9.2	-69.1
1.1	.35	6.7	9.7	-77.6
1.2	.35	6.7	10.2	-85.9

The next computation is JSTOP which is one of the stopping criteria. JSTOP is computed by taking all components of the gradient, squaring them, summing them together and then taking the square root. As the gradient diminishes in magnitude after each iteration so must the JSTOP value. Once directions of objective functional minimization have been established, the algorithm computes the single best step size to proceed in. GLAMB computes three objective functional values for three step sizes used in the algorithm. The three step sizes are 0, .5 and 1. Using Eq (3.29) a new maneuver vector is computed using each of the three step sizes. Each new maneuver vector now has different speeds and banks in it. Each set of changed banks

and speeds are input into the objective functional and a value is computed. GLAMB will always have three values since there are three step sizes. The first value of GLAMB in Table 2 represents the objective functional value for a step size of 0 in the directions of the previously computed gradient. The second value of GLAMB is the objective functional value for a step size of .5 in the same direction and the last value a step size of 1. These three points are fit to a parabola and a minimum is established. The minimum value computed for this parabola is LAMBDA OPT, which is shown in Table 2 also. This is the best step size to use to derive a new maneuver vector. The final maneuver vector is computed using Eq (3.29) and LAMBDA OPT. The new maneuver vector is used to compute new direction gradients as well as a new JSTOP. This is one iteration for the algorithm. If LAMBDA OPT and JSTOP are small enough, the algorithm stops and prints the results. If not the whole process continues with computations of three new GLAMB's, a new LAMBDA OPT and finally a new final maneuver vector. Calculation and results for the example are shown after the first iteration, the 15th iteration and the final 27th iteration. Bank is output in radians, azimuth (theta) in degrees, speed in nm/min, and range in nautical miles. The iterations show how the CMMCA is gradually directed to take out the 20 degree bank and fly straight and level behind the CM.

Table 4. Calculations After 15th Iteration

DELTJ	JSTOP	GLAMB	LAMBDA OPT
-.09804743	5.328728	6.364915	.08333333
-.1028940		6.111893	
-.1088278		5.858871	
-.1154768			
-.1222782			
-.1280745			
-.1309998			
-.1286812			
-.1179303			
-.09494756			
-.05623874			
-.01307128			
-.9609199			
-1.063918			
-1.179684			
-1.307672			
-1.445859			
-1.590090			
-1.733149			
-1.863540			
-1.964104			
-2.010684			
-1.971537			
-.4676461			

Table 5. Results After 15th Iteration

TIME	BANK	SPEED	RANGE	THETA
.1	-.0	6.7	8.0	.2
.2	-.0	6.7	8.0	.7
.3	-.0	6.7	8.0	1.4
.4	-.0	6.7	8.0	2.2
.5	-.1	6.7	8.0	3.2
.6	-.1	6.7	8.0	4.5
.7	-.1	6.7	8.0	6.3
.8	-.1	6.7	8.0	8.5
.9	-.1	6.7	8.0	11.3
1.0	-.2	6.7	8.0	14.7
1.1	-.2	6.7	8.1	18.9
1.2	-.1	6.7	8.1	22.6

Conclusion

A relatively simple objective functional has been defined for the CMMCA problem. A modified gradient search procedure written in FORTRAN code will solve for the optimal banks and airspeeds given a particular cruise missile maneuver. An initial CMMCA starting search vector is being provided by a previously written simulation model which outputs cruise missile bank and airspeed throughout a maneuver. The overall implementation process was discussed tying all parts of the solution process and evaluation together. Finally, an example of how the algorithm computes

a more optimal bank and speed profile for the CMMCA given a cruise missile flight path was discussed.

Table 6. Calculations After 27th Iteration

DELTJ	JSTOP	GLAMB	LAMBDA OPT
.0117851	.05944142	5.049137	.02235206
.01140183		5.0623425	
.0181835		5.075548	
.01015037			
.0093160			
.0083281			
.0072930			
.00611111			
.0047750			
.003370386			
.0017806			
.000427			
-.0128183			
-.015597			
-.0179451			
-.0196714			
-.020569			
-.0204229			
-.0190244			
-.016200			
-.0118509			
-.0060088			
-.00110236			
-.00116237			

Table 7. Final Results 27th Iteration

TIME	BANK	SPEED	RANGE	THETA
.1	.0	6.7	8.0	.0
.2	.0	6.7	8.0	.0
.3	.0	6.7	8.0	.0
.4	.0	6.7	8.0	.1
.5	.0	6.7	8.0	.1
.6	.0	6.7	8.0	.1
.7	.0	6.7	8.0	.2
.8	.0	6.7	8.0	.2
.9	.0	6.7	8.0	.2
1.0	.0	6.7	8.0	.3
1.1	.0	6.7	8.0	.3
1.2	.0	6.7	8.0	.3

IV. Results

A guidance algorithm was successfully built and tested to help determine the optimal controls required to track a cruise missile by CMMCA. Modifications of the model were made along the way to make it faster, more efficient and reliable. Memory requirements were kept to a minimum to avoid problems encountered by Heavner. The next few sections describe some of the algorithm problems and fixes, operational validation efforts and actual turn results.

Problems

To ensure the methodology was feasible a smaller test objective function with a known answer was run through the optimization algorithm. As expected the test case ran perfectly with only minor adjustments required to achieve the accuracy desired.

The next step was to customize the objective functional for the CMMCA problem into the optimization algorithm. A small simple maneuver profile was used to verify and validate the algorithm. One of the first test runs showed the optimal true airspeeds were cycling back and forth with variances as large as 20 knots. This was completely unanticipated and unacceptable as an answer. The apparent cause for the cycling was the integration weight approximation scheme being used. Instead of using the

simple trapezoidal rule as mentioned in the methodology section a more sophisticated and accurate Simpson's rule was applied (7:146). When the simpler trapezoidal rule was used in place of the more complex Simpson's rule the cycling disappeared.

The next problem occurred with the parabola fitting technique used to compute the optimal step size. Every so often a negative step size was computed stepping the algorithm in the opposite direction of minimization. The fix was to use a simpler more efficient technique for fitting the parabola. Instead of using three points to fit the parabola, two points and the slope at one of these points was used. Just like the three point parabola fitting case, one of the points is at the origin along with the slope and the other is some step size away. The new curve fitting procedure did not involve any extra computations and actually saved one more step size computation. The slope was already being computed through the gradient so it was already available. Derivation of the new curve fitting technique follows the three point procedure in Appendix C.

As confidence built in running the algorithm it was noticed that the computed optimal step size was consistently much smaller than the 1 and .5 initially input. There was noticeable improvement in the objective functional using step sizes much smaller than the guessed values. With

analysis into Eq (3.29) it became obvious the initial lambda's or step sizes were too large. Eq (3.29) shows how a new more optimal maneuver vector is computed using the gradient and step size. The maneuver vector has units of nm/min for the speeds and radians for the bank. If the gradient says to move one unit in the opposite direction some length, this is subtracted from the old vector and becomes the new more optimal vector. The original step sizes were 1 and .5. This means moving 1 nm/min in speed and 1 radian in bank. That equates to 60 knots in speed and 57 degrees of bank for the step size of 1 and half of this for the .5 step size. These values are much too large to move in one iteration. To alleviate the problem the lambda's were scaled to represent 5 knots change in speed and 5 degrees in bank. This equated to using a lambda of approximately .08333. With the new curve fitting approach the two lambda's required would be 0 and .08333. The optimal step size computed from these initial two values was now more consistent with the units being used for each iteration.

Validation

After the major problems were alleviated small test runs were accomplished to help validate the model. Two small maneuver vectors, consisting of 12 banks and 12 speeds for a total maneuver time of 1.2 minutes, were used to test

the algorithm's responsiveness. The first profile had the cruise missile flying straight and level while the CMMCA was placed directly behind at the desired nominal location and speed; 8 miles range, 0 degrees azimuth, and 400 knots true airspeed (TAS). The algorithm responded with a no change policy leaving the original bank and speed vectors untouched. The input information and results appear in Tables 8 and 9 on the next page. The columns marked xcm and ycm are the x and y position of the cruise missile.

The second small profile consisted of the same straight and level course for the cruise missile while the CMMCA was placed directly behind as before but with a 20 degree bank. As was expected the optimal control was to take the bank immediately out and continue flying straight and level behind the CM. Tables 10 and 11 show the input and results for this profile. These short profiles took a matter of minutes to run, both on the PC and the VAX.

Longer more realistic maneuvers were chosen next to further ensure model validity. The four selected profiles tested the algorithm's reaction to a cruise missile flying straight and level at a nominal speed of 400 knots TAS while the CMMCA was placed at different locations and speeds. The first location was 5 nm to the left of the cruise missile a slant range distance of 9.5 nm behind at 400 knots TAS. The algorithm results said to bank right approximately 30

degrees decreasing as the azimuth to the cruise missile diminished to zero. CMMCA speed was initially increased until the 9.5 nm range was cut to 8 nm. Figure 4 depicts the results graphically.

The next profile the CMMCA was located 5 nm to the right of the cruise missile at 400 TAS and 9.3 nm slant range. The optimal algorithm controls were similar to the preceding profile but in the opposite direction. The CMMCA banked left and sped up to align itself in the desired position, directly behind at 8 nm. Figure 5 graphically depicts this profile.

Table 8. Input Profile 1

TIME	BANK	SPEED	XCM	YCM
0.0	0.0	6.667	0.0	8.000
0.1	0.0	6.667	0.0	8.667
0.2	0.0	6.667	0.0	9.333
0.3	0.0	6.667	0.0	10.000
0.4	0.0	6.667	0.0	10.667
0.5	0.0	6.667	0.0	11.333
0.6	0.0	6.667	0.0	12.000
0.7	0.0	6.667	0.0	12.667
0.8	0.0	6.667	0.0	13.333
0.9	0.0	6.667	0.0	14.000
1.0	0.0	6.667	0.0	14.667
1.1	0.0	6.667	0.0	15.333
1.2	0.0	6.667	0.0	16.000

Table 9. Results Profile 1

# ITERATIONS = 1				
TIME	BANK	SPEED	RANGE	THETA
0.0	0.0	400.0	8.0	0.0
0.2	0.0	400.0	8.0	0.0
0.3	0.0	400.0	8.0	0.0
0.4	0.0	400.0	8.0	0.0
0.5	0.0	400.0	8.0	0.0
0.6	0.0	400.0	8.0	0.0
0.7	0.0	400.0	8.0	0.0
0.8	0.0	400.0	8.0	0.0
0.9	0.0	400.0	8.0	0.0
1.0	0.0	400.0	8.0	0.0
1.1	0.0	400.0	8.0	0.0
1.2	0.0	400.0	8.0	0.0

The third profile demonstrated the response if speeds did not match. The CMMCA was to the left and behind the cruise missile as before. This time however, the CMMCA was flying a slower 360 knots TAS. The algorithm corrected the offset error as before and sped up to match the cruise missile speed of 400 knots TAS. Once it had reached the desired location however it allowed the airspeed to start drifting back to the original 360 knots. There may be something within the objective functional driving the algorithm optimal results to return to the initial conditions at the end of a maneuver. Allowing 750

iterations took the VAX system about one hour of CPU time for each of the validation tests.

Table 10. Input Profile 2

TIME	BANK	SPEED	XCM	YCM
0.0	0.00	6.667	0.0	8.000
0.1	0.35	6.667	0.0	8.667
0.2	0.35	6.667	0.0	9.333
0.3	0.35	6.667	0.0	10.000
0.4	0.35	6.667	0.0	10.667
0.5	0.35	6.667	0.0	11.333
0.6	0.35	6.667	0.0	12.000
0.7	0.35	6.667	0.0	12.667
0.8	0.35	6.667	0.0	13.333
0.9	0.35	6.667	0.0	14.000
1.0	0.35	6.667	0.0	14.667
1.1	0.35	6.667	0.0	15.333
1.2	0.35	6.667	0.0	16.000

The last profile, depicted in Figure 6, put the CMMCA in front of and to the right of the cruise missile. The cruise missile was flying the same straight flight path as before. The algorithm attempted to place the CMMCA in front of the cruise missile at the 8 nm point. The gradient did not come close to vanishing and cycled between two values the entire 100 iterations. It makes sense that the gradient value would remain large since the objective functional was

still penalizing for the azimuth being so high. Appendix D contains tabular output from these runs as well as the remaining turn profiles.

Table 11. Results Profile 2

# ITERATIONS = 41				
TIME	BANK	SPEED	RANGE	THETA
.1	.0	400.6	8.0	.0
.2	.0	400.6	8.0	.0
.3	.0	400.5	8.0	.0
.4	.0	400.5	8.0	.0
.5	.0	400.4	8.0	.0
.6	.0	400.3	8.0	.0
.7	.0	400.2	8.0	.0
.8	.0	400.1	8.0	.0
.9	.0	400.0	8.0	.0
1.0	.0	400.0	8.0	.0
1.1	.0	399.9	8.0	.0
1.2	1.0	399.9	8.0	-.1

Turns

Two true turn profiles were attempted to learn more about the objective functional and ascertain any more difficulties with the methodology besides the occasional speed or bank parameter exceeding aircraft capabilities. The first turn profile the cruise missile flew a 90 degree right turn while in 20 degrees of bank. The CMMCA was started at the desired position of 8 nm directly behind the

missile at the matching speed of 400 knots TAS. The algorithm had no problem allowing the CMMCA to track the CM without exceeding any aircraft flight parameters. The second cruise missile profile consisted of one 20 degree bank, 270 degree turn at 400 knots TAS. The CMMCA started in the desired location directly behind the cruise missile at 8 nm at a speed of 400 knots. Figure 7 shows the results. The algorithm allowed the CMMCA to exceed tolerances in bank several times. This possibly could be the best the algorithm could do to keep the cruise missile within the radar cone limits.

Summary

Several runs were used to test the algorithm's operational validity and verify the computer results. One run demonstrated the algorithm's difficulty with the CMMCA flying in a position ahead of the CM. As shown by the turns, the algorithm responds fairly well in the types and sizes of control inputs that are required to improve the flight path of the CMMCA in a specific maneuver.

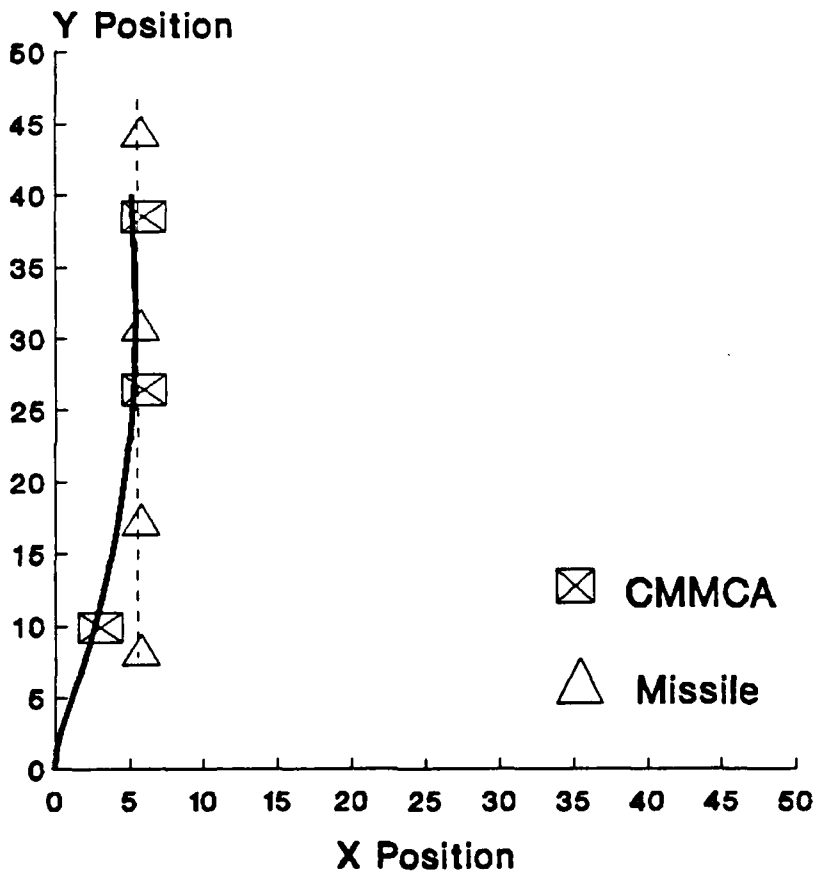

Figure 4. CMMCA Left and Behind CM

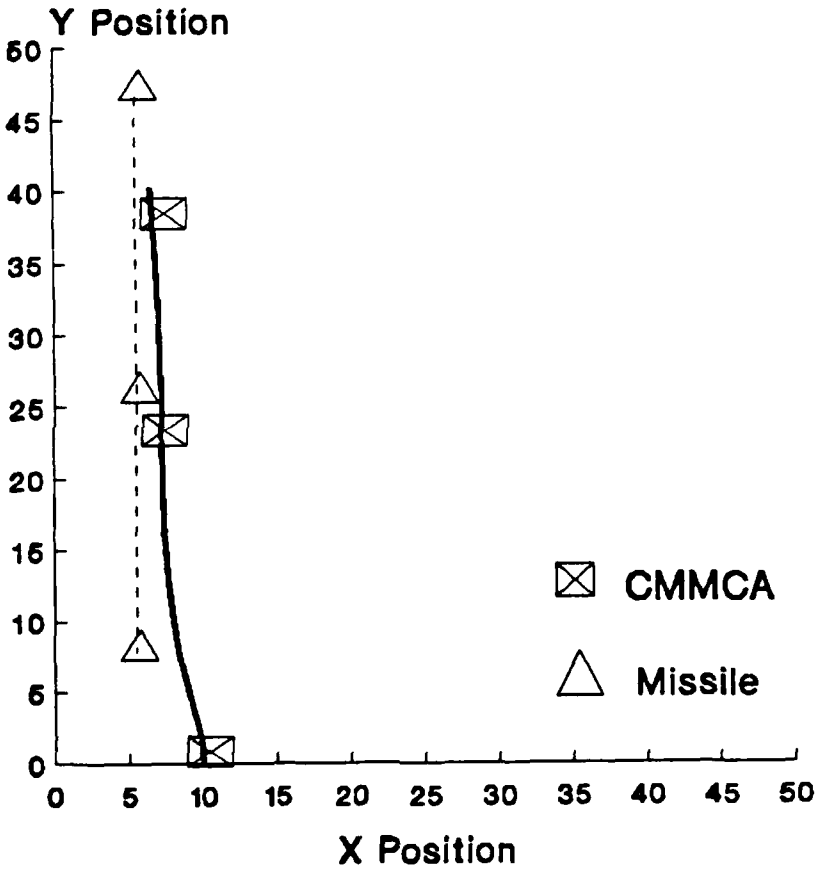

Figure 5. CMMCA Right and Behind CM

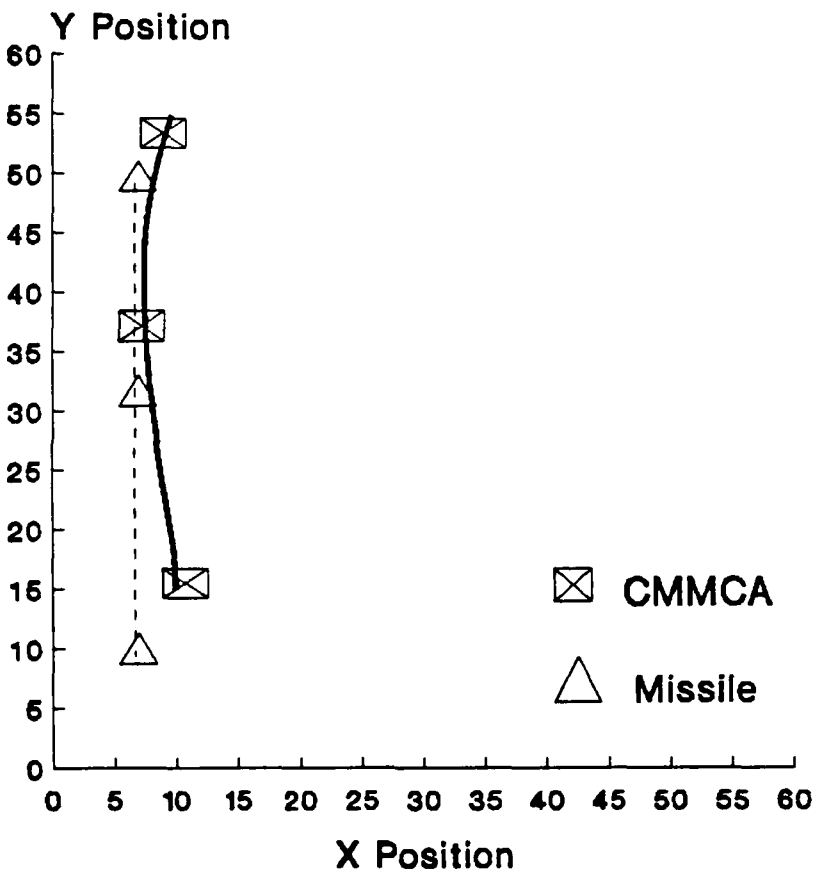

Figure 6. CMMCA Right and in Front of CM

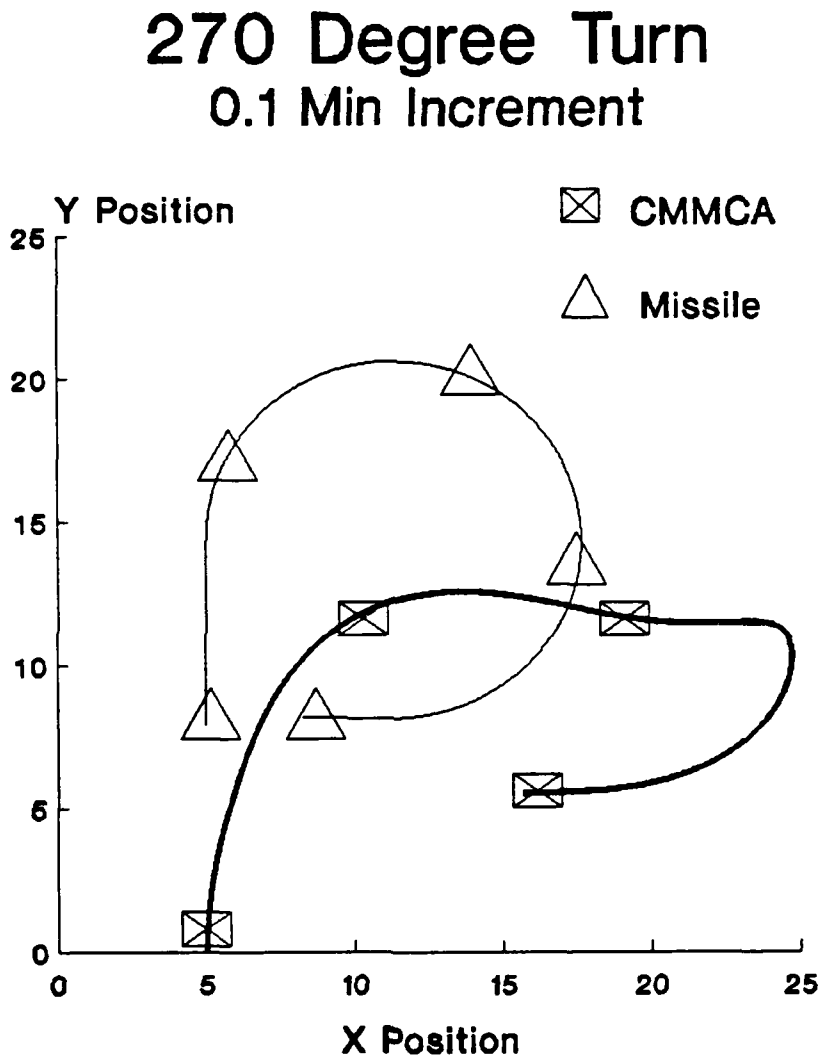

Figure 7. 270 Degree Turn

V. Conclusions and Recommendations

Conclusions

Using CMMCA and its new radar system to track a cruise missile is an extremely difficult and complex problem. The primary purpose of this thesis was to identify an algorithm that could solve for an optimal CMMCA flight path given a cruise missile flight maneuver. The simplest formulation for an objective functional was established taking into account performance constraints as well as desired position constraints.

Minimizing the objective functional without being able to take the second derivative was one of the greatest obstacles. In order for the algorithm to attain optimality the gradient must vanish or come within some reasonable criteria of zero.

The objective functional formulated has many variables that can be manipulated and adjusted. None of these adjustments have been properly analyzed for their overall influence on the way the algorithm optimizes the CMMCA flight path. It is believed that adjusting these weights can help alleviate exceeding CMMCA flight parameters.

An advantage of the solution method is its continued potential for use on a PC so crews can use it for mission planning. Excessive memory storage is not required and

keeping the profile down to a single maneuver should allow for reasonable response times.

Recommendations

The technique used still appears a good possibility in providing crews with valuable information on where to fly to minimize deviations in tracking the cruise missile.

The optimization algorithm needs to be tested under more conditions and situations that may be encountered. A larger cross section of the possible cruise missile flight paths should be tested against the algorithm to help detect any problems. Multiple maneuvers as well as longer duration maneuvers could be studied also.

Analysis should be done on the adjustable variables in the objective functional equation. Examination of these variables is important because they may have a dramatic effect on what the algorithm computes as optimal for the same maneuver.

Another important parameter needing further study is the time increment between control inputs. Is the time increment of a tenth of a minute used in the study appropriate for the problem at hand? Heavner found different time intervals had quite an impact on the effectiveness of his performance criteria. There is a good

possibility the same will apply for the optimization algorithm in this thesis.

Finally, if the guidance algorithm product is deemed suitable for use, increased effort should be made to make the algorithm as efficient, fast, and user friendly as possible so it can be used with confidence by the crews who so desperately need it to mission plan.

Appendix A. Equations of Motion

The equations of motion set forth here will apply to the CMMCA only. The axis of reference for these equations will be the x-axis representing east and the y-axis representing north. Heading, Hdg, is defined as the angle between the y-axis and the velocity vector measured in a clockwise direction. Heading can range from 0 to 360 degrees. The rate of change for the x position, y position and heading can now be derived.

State Equations

According to basic physics, the x-y components of the velocity vector define the rate of change of the aircraft's x and y position. When an aircraft is flying a heading, Hdg in degrees, and a true airspeed, TAS in knots, the x-y components can be written as:

$$V_x = TAS \sin[Hdg] \tag{A.1}$$

$$V_y = TAS \cos[Hdg] \tag{A.2}$$

The wind vector, which can also be written in its two x-y components, can be added to the above equations using vector algebra. With a notation change to include time the following equations define the x and y position rate of change for the CMMCA.

$$\frac{d\,x(t)}{dt} = WV_x + TAS(t)\,\sin[Hdg(t)] \qquad \text{(A.3)}$$

$$\frac{d\,y(t)}{dt} = WV_y + TAS(t)\,\cos[Hdg(t)] \qquad \text{(A.4)}$$

In a level turn, the heading rate of change is defined as

$$\frac{d\,Hdg(t)}{dt} = g\,\frac{\tan[\alpha(t)]}{TAS(t)} \qquad \text{(A.5)}$$

where g is the magnitude of the acceleration due to gravity in nm per minute squared and alpha is the particular bank angle in radians the CMMCA is in at a particular time. The initial conditions for CMMCA position and heading will be needed in order to integrate Eqs (A.3), (A.4) and (A.5).

$$x(t_0) = x_0$$
$$y(t_0) = y_0$$
$$Hdg(t_0) = Hdg_0$$

Integrating the equations using the preceding initial conditions yields:

$$x(t) = x_0 + \int_{t_0}^{t} TAS(\tau)\sin[Hdg(\tau)]\,d\tau + WV_x(t-t_0) \qquad \text{(A.6)}$$

$$y(t) = y_0 + \int_{t_0}^{t} TAS(\tau)\cos[Hdg(\tau)]\,d\tau + WV_x(t-t_0) \qquad \text{(A.7)}$$

$$Hdg(t) - Hdg_0 + \int_{t_0}^{t} g \frac{\tan[\alpha(\tau)]}{TAS(\tau)} d\tau \qquad \text{(A.8)}$$

In order to use these equations in a computer code utilizing numerical methods for solution one must discretize the time points. Discretizing the above integrals adds a term called an integration weight represented by W_1. Each integration weight helps approximate the integral that has been replaced by a summation sign. The trapezoidal rule will be used to supply the numerical weights for these three integrations. The final form for the equations of motion are:

$$x_i - x_0 + \sum_{I=0}^{i} TAS_I \sin[Hdg_I] W_I^{(i)} \Delta t + WV_x(t - t_0) \qquad \text{(A.9)}$$

$$y_i - y_0 + \sum_{I=0}^{i} TAS_I \cos[Hdg_I] W_I^{(i)} \Delta t + WV_x(t - t_0) \qquad \text{(A.10)}$$

$$Hdg_I - Hdg_0 + \sum_{s=0}^{I} g \frac{\tan[\alpha_s]}{TAS_s} W_s^{(I)} \Delta t \qquad \text{(A.11)}$$

These equations completely describe the motion of the CMMCA given an initial position and heading along with a bank angle and true airspeed for each time increment under consideration.

Appendix B. Source Code

The source code for the FORTRAN program used to implement the guidance algorithm follows. The code was compiled and executed on a 386 PC using Microsoft's FORTRAN 77 compiler. The code was also compiled and executed on a VAX running on the VMS operating system.

Following the guidance algorithm is a SLAM II simulation code. SLAM was used to generate the cruise missile's x-y position during a maneuver and to generate an initial guess for CMMCA bank and speed as inputs in the guidance algorithm.

```
C*********************************************************************
C*                                                                   *
C*  TITLE:    CMMCA Program to Track Cruise Missile Maneuvers        *
C*                                                                   *
C*  AUTHOR:   Capt Tony M. Garton                                    *
C*  DATE:     11 Feb 90                                              *
C*                                                                   *
C*  DESCRIPTION:  This program implements an algorithm to track      *
C*                a cruise missile during a turning maneuver.        *
C*                The program is not user friendly but can easily    *
C*                be learned and used on any IBM PC that has a       *
C*                FORTRAN compiler available.                        *
C*********************************************************************

      PROGRAM CMMCA

      INTEGER I,LIM,ALL,ITER,K3,K4
      PARAMETER (LIM=12,ALL=24)
      REAL DELTJ(1:ALL),GLAMB(0:1),VLAMB(1:ALL,0:1),
     &     BANK(0:LIM),SPEED(0:LIM),XCM(0:LIM),YCM(0:LIM),
     &     XPLANE(0:LIM),YPLANE(0:LIM),RANGE(1:LIM),THETA(1:LIM),
     &     HDG(0:LIM),WIND(1:2),LAMBDA(0:1),VECTOR(1:ALL),NEWV(1:ALL),
     &     QUADW(0:LIM),W(1:3),WK(0:LIM,1:LIM)
      REAL LAMBOP,JSTOP,DT,R0,THETA0,MU,U0,B0
      COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG
      COMMON /WEIGHT/ QUADW,W,WK,K3,K4
      COMMON /PRELIM/ R0,THETA0,MU,U0,B0

C  Time increment in minutes for the algorithm
      DT=.1

      CALL INPUT(VECTOR,LAMBDA,WIND,DT)
      CALL DELTAJ(DELTJ,JSTOP,WIND,DT)
10    CALL VLAMDA(VECTOR,LAMBDA,VLAMB,DELTJ)
      CALL GLAMDA(VLAMB,GLAMB,WIND,DT)
      CALL OPT(LAMBDA,GLAMB,LAMBOP,JSTOP)
      CALL NEWPOS(VECTOR,LAMBOP,DELTJ,NEWV)
      CALL DECOMP(NEWV)
      CALL DELTAJ(DELTJ,JSTOP,WIND,DT)
      CALL CHECK(LAMBOP,JSTOP,NEWV,VECTOR)
      GOTO 10
      END
```

```
C*********************************************************************
C*                                                                   *
C*   SUBROUTINE:    INPUT                                            *
C*                                                                   *
C*   DESCRIPTION:   Used to input necessary data for running the     *
C*                  algorithm.                                       *
C*********************************************************************

      SUBROUTINE INPUT(VECTOR,LAMBDA,WIND,DT)
      INTEGER I,J,LIM,ALL,K3,K4
      PARAMETER (LIM=12,ALL=24)
      REAL DT,R0,THETA0,MU,U0,B0
      REAL VECTOR(1:ALL),LAMBDA(0:1),WIND(1:2),BANK(0:LIM),
     &     SPEED(0:LIM),XCM(0:LIM),YCM(0:LIM),XPLANE(0:LIM),
     &     YPLANE(0:LIM),RANGE(1:LIM),THETA(1:LIM),HDG(0:LIM),
     &     QUADW(0:LIM),W(1:3),WK(0:LIM,1:LIM)
      COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG
      COMMON /WEIGHT/ QUADW,W,WK,K3,K4
      COMMON /PRELIM/ R0,THETA0,MU,U0,B0

C All speeds are in NM/MIN and all angles are in radians.
C Initial heading and x-y position for the CMMCA

      HDG(0)=0
      XPLANE(0)=0
      YPLANE(0)=0

C Read in a vector containing the initial guess of bank and speed
C CMMCA should be in during the maneuver.  Also read in x-y position
C for the cruise missile for each time increment of the maneuver.

      OPEN(10,FILE='INPUT.DAT',STATUS='OLD')
      DO 10 I=0,LIM
        READ(10,*) BANK(I),SPEED(I),XCM(I),YCM(I)
10    CONTINUE
      CLOSE(10)

C Contains the step size for the gradient search

      OPEN(11,FILE='LAMBDA.DAT',STATUS='OLD')
      READ(11,*) (LAMBDA(I), I=0,1)
      CLOSE(11)

C Contains nominal values of range, azimuth, allowable speed variance,
C speed,and bank angle variance.

      OPEN(12,FILE='NOMINAL.DAT',STATUS='OLD')
      READ(12,*) R0,THETA0,MU,U0,B0
      CLOSE(12)
```

```
C Wind vector in its two x-y components.

         OPEN(13,FILE='WIND.DAT',STATUS='OLD')
         READ(13,*) (WIND(I), I=1,2)
         CLOSE(13)

C Adjustable objective function weights.

         OPEN(14,FILE='WEIGHT.DAT',STATUS='OLD')
         READ(14,*) (W(I), I=1,3)
         CLOSE(14)

C Adjustable parabola weights.

         OPEN(15,FILE='K.DAT',STATUS='OLD')
         READ(15,*) K3,K4
         CLOSE(15)

C Initialize integration weight matrix.

         DO 30 I=0,LIM
            DO 20 J=1,LIM
                  WK(I,J)=0
20          CONTINUE
30       CONTINUE

C Read in starter integration weight matrix.  Used the trapezoidal rule.

         OPEN(16,FILE='WK.DAT',STATUS='OLD')
           DO 40 I=0,3
              READ(16,*) (WK(I,J), J=1,3)
40         CONTINUE
           CLOSE(16)

           DO 60 I=0,3
               DO 50 J=1,3
                    WK(I,J)=WK(I,J)*DT
50             CONTINUE
60         CONTINUE

C Generates the full integration weight matrix depending on how long
C the maneuver is.

           CALL GEN(WK)

C Takes the last column of integration weight matrix and uses it for
C the quadrature objective function weights.

         DO 70 I=0,LIM
```

```
            QUADW(I)=WK(I,LIM)/DT
70        CONTINUE

C Combines inputted speed and bank into one long vector for later
C use in the optimization routine.

         DO 80 I=1,LIM
            VECTOR(I)=SPEED(I)
            VECTOR(LIM+I)=BANK(I)
80       CONTINUE
         END

C*********************************************************************
C*                                                                   *
C*   SUBROUTINE:     GEN                                             *
C*                                                                   *
C*   DESCRIPTION:    Generates the full integration weight matrix    *
C*                   from the inputted starter matrix.  The size     *
C*                   of the generated matrix depends on the          *
C*                   maneuver length.                                *
C*                                                                   *
C*********************************************************************

          SUBROUTINE GEN(WK)

          INTEGER LIM,ROW,COL,X,Y,I,COUNT
          PARAMETER (LIM=12)
          REAL WK(0:LIM,1:LIM)
          X=2
          Y=2
          COUNT=0

          DO 20 COL=4,LIM
              I=1
              DO 10 ROW=0,LIM
                   IF (ROW .LT. X) THEN
                       WK(ROW,COL)=WK(ROW,X)
                   ENDIF
                   IF (ROW .EQ. X) THEN
                       WK(ROW,COL)=WK(ROW,X) + WK(0,Y)
                   ENDIF
                   IF (ROW .GT. X) THEN
                       WK(ROW,COL)=WK(I,Y)
                       I=I+1
                   ENDIF
10            CONTINUE
              COUNT = COUNT + 1
              IF (Y .EQ. 2) THEN
                   Y=3
```

```
                ELSE
                    Y=2
                ENDIF
                IF (COUNT .EQ. 2)THEN
                    X=X + 2
                    COUNT=0
                ENDIF
20          CONTINUE
            END


C*****************************************************************
C*                                                               *
C*    SUBROUTINE:     DECOMP                                     *
C*                                                               *
C*    DESCRIPTION:    Decomposes the maneuver vector back into its *
C*                    original bank and speed vectors.           *
C*                                                               *
C*****************************************************************

            SUBROUTINE DECOMP(V)

            INTEGER I,LIM,ALL
            PARAMETER (LIM=12,ALL=24)
            REAL V(1:ALL),BANK(0:LIM),SPEED(0:LIM)
            COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG

            DO 10 I=1,LIM
                SPEED(I)=V(I)
                BANK(I)=V(LIM+I)
10          CONTINUE
            END

C*****************************************************************
C*                                                               *
C*    SUBROUTINE:     COMP                                       *
C*                                                               *
C*    DESCRIPTION:    Computes CMMCA position and heading given  *
C*                    bank and speed vectors for the maneuver.   *
C*                    Also computes range and azimuth from the   *
C*                    CMMCA to the CM given the x and y position *
C*                    for the CM for the entire maneuver.        *
C*                                                               *
C*****************************************************************

            SUBROUTINE COMP(WIND,DT)

            INTEGER I,LIM,S,K3,K4
            REAL DT,SUM1,SUM2,SUM3,TWOPI,ALPHA
```

```
      PARAMETER (LIM=12)
      REAL  BANK(0:LIM),SPEED(0:LIM),XCM(0:LIM),YCM(0:LIM),
     &      XPLANE(0:LIM),YPLANE(0:LIM),RANGE(1:LIM),THETA(1:LIM),
     &      HDG(0:LIM),QUADW(0:LIM),W(1:3),WK(0:LIM,1:LIM)
     &      WIND(1:2)
      COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG
      COMMON /WEIGHT/ QUADW,W,WK,K3,K4

      TWOPI=2*(ACOS(-1.))
      SUM1=0
      SUM2=0
      SUM3=0
      DO 40 I=1,LIM
         DO 10 S=0,I
            SUM1=SUM1 + WK(S,I)*19.05*TAN(BANK(S))/SPEED(S)
10       CONTINUE
         HDG(I)=HDG(0) + SUM1

C Ensures heading lies between 0 and 2 pi.

20          IF (HDG(I) .GT. TWOPI)THEN
               HDG(I)=HDG(I)-TWOPI
               GOTO 20
            ENDIF
30          IF (HDG(I) .LT. -TWOPI)THEN
               HDG(I)=HDG(I)+TWOPI
               GOTO 30
            ENDIF
         SUM1=0
40    CONTINUE
      DO 60 I=1,LIM
         DO 50 S=0,I
            SUM2=SUM2 + SPEED(S)*SIN(HDG(S))*WK(S,I)
            SUM3=SUM3 + SPEED(S)*COS(HDG(S))*WK(S,I)
50       CONTINUE
         XPLANE(I)=XPLANE(0) + SUM2 +  WIND(1)*DT*I
         YPLANE(I)=YPLANE(0) + SUM3 +  WIND(2)*DT*I
         SUM2=0
         SUM3=0
60    CONTINUE
      DO 70 I=1,LIM
         RANGE(I)=SQRT((XCM(I)-XPLANE(I))**2 +
     &   (YCM(I)-YPLANE(I))**2)
C  The angle ALPHA is a very important measure.  The angle is
C  measured from the North or y axis to the CM.  Counterclockwise
C  from the y axis is a negative ALPHA.  Measuring clockwise from
C  the y axis is a positive angle.  Because of the TAN2 function
C  ALPHA only ranges from pi to -pi.
```

```
            ALPHA= ATAN2(XCM(I)-XPLANE(I),YCM(I)-YPLANE(I))
            THETA(I)= ALPHA - HDG(I)

C Ensures theta remains between -pi and pi.

            IF (THETA(I) .LT. -TWOPI/2) THETA(I)=THETA(I) + TWOPI
70          CONTINUE
            END

C*********************************************************************
C*                                                                   *
C*    SUBROUTINE:     DELTAJ                                         *
C*                                                                   *
C*    DESCRIPTION:    Computes the gradient of J with respect to     *
C*                    velocity and bank angle.                       *
C*                                                                   *
C*                                                                   *
C*********************************************************************

            SUBROUTINE DELTAJ(DELTJ,JSTOP,WIND,DT)

            INTEGER I,K,K3,K4,KPL
            PARAMETER (LIM=12,ALL=24)
            REAL  BANK(0:LIM),SPEED(0:LIM),XCM(0:LIM),YCM(0:LIM),
     &          XPLANE(0:LIM),YPLANE(0:LIM),RANGE(1:LIM),THETA(1:LIM),
     &          HDG(0:LIM),WIND(1:2),QUADW(0:LIM),W(1:3),WK(0:LIM,1:LIM)
     &          DELTJ(1:ALL)
            REAL JSTOP,R0,THETA0,MU,U0,B0,DT,TEMP,A,B,C,L,M,
     &          N,O,P,Q,R,S,T,X,Y,Z,SUM,SUM1,SUM2,SUM3,SUM4
            COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG

            COMMON /PRELIM/ R0,THETA0,MU,U0,B0
            COMMON /WEIGHT/ QUADW,W,WK,K3,K4

            SUM=0
            SUM1=0
            SUM2=0
            SUM3=0
            SUM4=0
            CALL COMP(WIND,DT)

C Initializes gradient vector to zero.

            DO 10 I=1,ALL
               DELTJ(I)=0
10          CONTINUE
            DO 40 K=1,LIM
               KPL=K+LIM
               DO 30 I=1,LIM
```

```
                    M=(RANGE(I)-R0)
                    N=(THETA(I)-THETA0)
                    S=(XCM(I)-XPLANE(I))
                    T=(YCM(I)-YPLANE(I))

               IF (K .LE. I)THEN

                    IF (I .EQ. K)THEN
                         SUM1=(W(2)*2*K3/MU)*(((SPEED(I)-U0)/MU)
     &                    **(2*K3-1))
                         SUM2=(W(3)*2*K4/B0)*((BANK(I)/B0)**(2*K4-1))
                    ENDIF
               X=-SIN(HDG(K))*WK(K,I)
               Y=-19.05*TAN(BANK(K))/(SPEED(K)**(2))
               Z=-COS(HDG(K))*WK(K,I)
               A=-19.05*((1/COS(BANK(K)))**2)/SPEED(K)
               DO 20 L=K,I
                    Q=SPEED(L)*COS(HDG(L))*WK(L,I)*WK(K,L)
                    R=SPEED(L)*SIN(HDG(L))*WK(L,I)*WK(K,L)
                    SUM3= SUM3 + Q
                    SUM4= SUM4 + R
20             CONTINUE
               O=(S*(X-(Y*SUM3)))+(T*(Z+(Y*SUM4)))
               P=(((-S*(Z+(Y*SUM4)))+(T*(X-(Y*SUM3))))/(S**2+T**2))-(Y
     &          *WK(K,I))
               SUM1=SUM1 + (2*M*O/RANGE(I)) + (2*W(1)*P)

               B=(S*(A*SUM3))+(T*(-A)*SUM4)
               C=(((-S*((-A)*SUM4))+(T*A*SUM3))/(S**2+T**2))+(A*WK(K,I))
               SUM2=SUM2 + (2*M*B/RANGE(I)) + (2*W(1)*N*C)
               ENDIF

C Sums the gradient vector for velocity.

               DELTJ(K)=DELTJ(K) + QUADW(I)*SUM1

C Sums the gradient vector for bank angle.

               DELTJ(KPL)=DELTJ(KPL) + QUADW(I)*SUM2
               SUM1=0
               SUM2=0
               SUM3=0
               SUM4=0
30          CONTINUE
40          CONTINUE

C Computes the slope of the gradient for each time increment.

            DO 50 I=1,ALL
```

```
            TEMP=DELTJ(I)**2
            SUM= SUM + TEMP
50          CONTINUE
            JSTOP=SQRT(SUM)

C Computes the normalized gradient of J.

            DO 60 I=1,ALL
               DELTJ(I)=DELTJ(I)/JSTOP
60          CONTINUE
            END

C*********************************************************************
C*                                                                   *
C*    SUBROUTINE:     VLAMDA                                         *
C*                                                                   *
C*    DESCRIPTION:    Computes a new column of the maneuver vector   *
C*                    for each of the two lambda step sizes.         *
C*                                                                   *
C*                                                                   *
C*********************************************************************

            SUBROUTINE VLAMDA(VECTOR,LAMBDA,VLAMB,DELTJ)

            INTEGER I,Q,ALL
            PARAMETER (ALL=24)
            REAL DELTJ(1:ALL),VLAMB(1:ALL,0:1),VECTOR(1:ALL),LAMBDA(0:1)
            DO 20 Q=0,1
               DO 10 I=1,ALL
                    VLAMB(I,Q)=VECTOR(I)-LAMBDA(Q)*DELTJ(I)
10             CONTINUE
20          CONTINUE
            END

C*********************************************************************
C*                                                                   *
C*    SUBROUTINE:     JCOMP                                          *
C*                                                                   *
C*    DESCRIPTION:    Computes a new objective function value        *
C*                    given appropriate inputs.                      *
C*                                                                   *
C*********************************************************************

            SUBROUTINE JCOMP(J)

            INTEGER I,LIM,K3,K4
            PARAMETER (LIM=12)
            REAL SUM,R0,THETA0,U0,MU,B0,J
            REAL QUADW(0:LIM),W(1:3),WK(0:LIM,1:LIM),BANK(0:LIM),
```

```
     &          SPEED(0:LIM),XCM(0:LIM),YCM(0:LIM),XPLANE(0:LIM),
     &          YPLANE(0:LIM),RANGE(1:LIM),THETA(1:LIM),HDG(0:LIM)
           COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG
           COMMON /PRELIM/ R0,THETA0,U0,MU,B0
           COMMON /WEIGHT/ QUADW,W,WK,K3,K4

           J=0
           DO 10 I=1,LIM
              SUM=(RANGE(I)-R0)**2 + W(1)*(THETA(I)-THETA0)**2
     &          + W(2)*(((SPEED(I)-U0)/MU)**(2*K3))
     &          + W(3)*((BANK(I)/B0)**(2*K4))
              J= J + QUADW(I)*SUM
10         CONTINUE
           END

C*********************************************************************
C*                                                                   *
C*    SUBROUTINE:     GLAMDA                                         *
C*                                                                   *
C*    DESCRIPTION:    Computes two new objective function values     *
C*                    for each of the new vectors generated in       *
C*                    the VLAMDA subroutine.  These two values       *
C*                    are evaluated in the OPT subroutine.           *
C*                                                                   *
C*********************************************************************

           SUBROUTINE GLAMDA(VLAMB,GLAMB,WIND,DT)

           INTEGER I,ALL,LIM,Q,K3,K4
           PARAMETER (LIM=12,ALL=24)
           REAL WIND(1:2),V(1:ALL),BANK(0:LIM),SPEED(0:LIM),
     &          XCM(0:LIM),YCM(0:LIM),XPLANE(0:LIM),YPLANE(0:LIM),
     &          RANGE(1:LIM),THETA(1:LIM),HDG(0:LIM),QUADW(0:LIM),
     &          W(1:3),WK(0:LIM,1:LIM),GLAMB(0:1),VLAMB(1:ALL,0:1)
           REAL J,DT,R0,THETA0,U0,MU,B0
           COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG
           COMMON /PRELIM/ R0,THETA0,U0,MU,B0
           COMMON /WEIGHT/ QUADW,W,WK,K3,K4

           DO 20 Q=0,1
              DO 10 I=1,ALL
                 V(I)=VLAMB(I,Q)
10            CONTINUE
              CALL DECOMP(V)
              CALL COMP(WIND,DT)
              CALL JCOMP(J)
              GLAMB(Q)=J
20         CONTINUE
           END
```

```
C*********************************************************************
C*                                                                   *
C*    SUBROUTINE:     OPT                                            *
C*                                                                   *
C*    DESCRIPTION:    Compares the two glamda values and computes    *
C*                    the best step size to take to a new and        *
C*                    better maneuver vector.                        *
C*                                                                   *
C*********************************************************************

          SUBROUTINE OPT(LAMBDA,GLAMB,LAMBOP,JSTOP)

          INTEGER LIM,ALL
          PARAMETER (LIM=12,ALL=24)
          REAL GLAMB(0:1),VLAMB(1:ALL,0:1),LAMBDA(0:1)
          REAL JSTOP,LAMBOP,Y


          Y=GLAMB(0) - JSTOP*LAMBDA(1)
          IF (GLAMB(1) .LE. Y) THEN
              LAMBOP=LAMBDA(1)

          ELSE
              LAMBOP=LAMBDA(1)*.5 + (.5*(GLAMB(1)-GLAMB(0))/(-JSTOP+
     &                ((GLAMB(0)-GLAMB(1))/LAMBDA(1))))
          ENDIF
          END

C*********************************************************************
C*                                                                   *
C*    SUBROUTINE:     NEWPOS                                         *
C*                                                                   *
C*    DESCRIPTION:    Computes the new maneuver vector given the     *
C*                    best step size determined in the previous      *
C*                    subroutine.                                    *
C*                                                                   *
C*********************************************************************

          SUBROUTINE NEWPOS(VECTOR,LAMBOP,DELTJ,NEWV)

          INTEGER I,ALL
          PARAMETER (ALL=24)
          REAL DELTJ(1:ALL),VECTOR(1:ALL),NEWV(1:ALL)
          REAL LAMBOP
          DO 10 I=1,ALL
              NEWV(I)= VECTOR(I)-LAMBOP*DELTJ(I)
10        CONTINUE
          END
```

```
C*********************************************************************
C*                                                                   *
C*    SUBROUTINE:     CHECK                                          *
C*                                                                   *
C*    DESCRIPTION:    Checks to see if the stopping criteria have    *
C*                    been met.  If they have output is generated.   *
C*                                                                   *
C*********************************************************************

          SUBROUTINE CHECK(LAMBOP,JSTOP,NEWV,VECTOR)

          INTEGER ITER,ALL,LIM,I
          PARAMETER (LIM=12,ALL=24)
          REAL JSTOP,LAMBOP
          REAL NEWV(1:ALL),VECTOR(1:ALL),BANK(0:LIM),SPEED(0:LIM),
     &         XCM(0:LIM),YCM(0:LIM),XPLANE(0:LIM),YPLANE(0:LIM),
     &         RANGE(1:LIM),THETA(1:LIM),HDG(0:LIM)
          COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG

          IF ((ABS(LAMBOP) .LT. .1).AND.(JSTOP .LT. .1)) THEN
              ITER=ITER + 1
              CALL OUTPUT(ITER)
          ELSE
              IF (ITER .EQ. 50) THEN
                   CALL OUTPUT(ITER)
              ENDIF
              ITER=ITER + 1

C Updates the old maneuver to the new maneuver vector.

              DO 10 I=1,ALL
                   VECTOR(I)=NEWV(I)
10            CONTINUE
          ENDIF
          END


C*********************************************************************
C*                                                                   *
C*    SUBROUTINE:     OUTPUT                                         *
C*                                                                   *
C*    DESCRIPTION:    Changes values of radians and NM/min back to   *
C*                    degrees and knots.  Also outputs the final     *
C*                    bank and speed vectors the CMMCA should fly.   *
C*                    Another file is generated to help graphically  *
C*                    look at the results.                           *
C*                                                                   *
C*********************************************************************
```

```
      SUBROUTINE OUTPUT(ITER)

      INTEGER I,ITER,LIM
      PARAMETER (LIM=12)
      REAL BANK(0:LIM),SPEED(0:LIM),XCM(0:LIM),YCM(0:LIM),
     &     XPLANE(0:LIM),YPLANE(0:LIM),RANGE(1:LIM),THETA(1:LIM),
     &     HDG(0:LIM)
      REAL A,B,C,D,PI
      COMMON BANK,SPEED,XCM,YCM,XPLANE,YPLANE,RANGE,THETA,HDG

      PI=ACOS(-1.)

      OPEN(21,FILE='RESULTS.OUT',STATUS='NEW')
      OPEN(22,FILE='PLOT.DAT',STATUS='NEW')

          WRITE(21,*) 'T     BANK     SPEED    RANGE    THETA'
          DO 10 I=1,LIM
               A=I*.1
               B=BANK(I)*180/PI
               C=SPEED(I)*60
               D=THETA(I)*180/PI
               WRITE(21,20) A,B,C,RANGE(I),D
10        CONTINUE
20        FORMAT(F3.1,4(3X,F6.1))

C File used by Harvard graphics to graphically display optimal
C CMMCA flight path compared to cruise missile flight path.

      DO 30 I=0,LIM
          WRITE(22,40) XPLANE(I),YPLANE(I),XCM(I),YCM(I)
30    CONTINUE
40    FORMAT(4(3X,F6.2))
      STOP
      END
```

```
;*************************************************************
;*                                                           *
;*                                                           *
;*    TITLE:  Cruise Missile Flight profile generator        *
;*                                                           *
;*    DESCRIPTION:  This SLAM II simulation program was      *
;*                  originally designed to heuristically     *
;*                  attempt to solve the CMMCA tracking      *
;*                  problem.  When a better approach than    *
;*                  heuristics was designed this program     *
;*                  was used primarily to output certain     *
;*                  cruise missile flight paths  as input    *
;*                  into the flight path optimization        *
;*                  routine.                                 *
;*                                                           *
;*************************************************************

GEN,GARTON,THESIS,12/1/90,1,N,N,,N,N,72;
LIMITS,0,2,50;
TIMST,XX(15),PERCENT IN CONE;
INIT,0,6;
CONTINUOUS,8,12,0.01,0.01,0.1,,.001,.001;

EQUIVALENCE/XX(1),ATASI/XX(2),ATH/XX(3),DCA/
           XX(5),WSPEED/XX(6),WDIR/XX(7),CTAS/
           SS(12),AGS/SS(1),CEPOS/SS(2),CNPOS/
           SS(3),CTC/SS(4),AEPOS;
EQUIVALENCE/SS(5),ANPOS/SS(6),ATC/SS(7),ATAS/
           SS(8),ABANGLE/SS(9),CBANGLE/DD(1),CEVEL/
           DD(2),CNVEL/DD(3),CTCC/DD(4),AEVEL/
           DD(5),ANVEL/DD(6),ATCC

RECORD,TNOW,MINUTES,20,T,0.1;
VAR,SS(9),,CBANGLE
VAR,XX(7),,CTAS
VAR,SS(1),,CEPOS;
VAR,SS(2),,CNPOS;

RECORD,TNOW,MINUTES,21,T,0.1;
VAR,SS(4),,AEPOS;
VAR,SS(5),,ANPOS;

;
;   DATA ENTRY CONVENTION
;
;   AIRSPEEDS IN KNOTS
;   TIMES IN MINUTES
;   ANGLES IN DEGREES
;   DISTANCES IN NAUTICAL MILES
;   HEADINGS REFERENCED TO NORTH AND INCREASE CLOCKWISE
```

```
;   BANK ANGLES TO THE RIGHT ARE POSITIVE IN VALUE
;
;   VARIABLE DESCRIPTION

;   XX(1)  - AIRCRAFT TAS INCREMENT
;   XX(2)  - AIRCRAFT TRUE HEADING
;   XX(3)  - DRIFT CORRECTION ANGLE (DCA)
;   XX(4)  - ACCELERATION DUE TO GRAVITY in NM/MIN**2
;   XX(5)  - WIND VELOCITY MAGNITUDE (FL280)
;   XX(6)  - WIND DIRECTION (FL280)
;   XX(7) - CRUISE MISSILE TAS
;   XX(9) - CONVERT CM HEADING TO RANGE OF 0-360
;   XX(10) - CONVERT CMMCA HEADING TO RANGE OF 0-360
;   XX(12) - INPUTED CMMCA ROLL RATE
;   XX(14) - IS CM INSIDE AZIMUTH
;   XX(15) - IS CM IN RADAR CONE
;   XX(16) - CM INSIDE RANGE
;   XX(17) - CMMCA MAX ROLL RATE (900. DEGREES/MINUTE)
;   XX(18) - INNER RANGE LIMIT
;   XX(19) - OUTER RANGE LIMIT
;   XX(20) - DESIRED CMMCA BANK ANGLE
;   XX(21) - SPEED MATCHING
;   XX(22) - POSITIVE REACTION AZIMUTH
;   XX(23) - NEGATIVE REACTION AZIMUTH

;   SS(1)  - CRUISE MISSILE EAST POSITION
;   SS(2)  - CRUISE MISSILE NORTH POSITION
;   SS(3)  - CRUISE MISSILE TRUE COURSE
;   SS(4)  - AIRCRAFT EAST POSITION
;   SS(5)  - AIRCRAFT NORTH POSITION
;   SS(6)  - AIRCRAFT TRUE COURSE
;   SS(7)  - AIRCRAFT INITIAL TAS
;   SS(8)  - AIRCRAFT BANK ANGLE
;   SS(9)  - CRUISE MISSILE BANK ANGLE
;   SS(10)  - AIRCRAFT/MISSILE SLANT RANGE
;   SS(11)  - AIRCRAFT/MISSILE AZIMUTH ANGLE
;   SS(12)  - AIRCRAFT GS

;   DD(1)  - CRUISE MISSILE EAST VELOCITY
;   DD(2)  - CRUISE MISSILE NORTH VELOCITY
;   DD(3)  - CRUISE MISSILE TRUE COURSE CHANGE RATE
;   DD(4)  - AIRCRAFT EAST VELOCITY
;   DD(5)  - AIRCRAFT NORTH VELOCITY
;   DD(6)  - AIRCRAFT TRUE COURSE CHANGE RATE
;   DD(7)  - ACCEL/DECEL RATE (A MAX OF .50/-.75 NM/MIN/MIN)
;   DD(8)  - CMMCA ROLL RATE (15 DEGREES/SEC, MAX)
;
INTLC, ATASI=0.0,ABANGLE=0.0,XX(4)=19.05,CBANGLE=0.0;
INTLC, CTAS=400.,CEPOS=5.0,CNPOS=8.0;
INTLC, CTC=0.0,ATC=0.0,XX(14)=1,XX(15)=1;
```

```
INTLC, XX(16)=1,XX(18)=2.,XX(19)=10.;

NETWORK;

; CM FLIGHT PATH

        CREATE;
        ACT,6.0;
        TERM,1;

;  CMMCA SPEED LIMITS (320-480 KNOTS TAS)

        DETECT,SS(7),XN,5.3333333,.0001;
        ASSIGN,ATASI = 0.,ATAS = 5.3333333;
        TERM;
        DETECT,SS(7),XP,8.,.0001;
        ASSIGN,ATASI = 0.,ATAS = 8.;
        TERM;

;    DETECT IF CM IS IN CMMCA RADAR CONE
;    XX(14), XX(15), XX(16) ARE FLAGS WHERE 1 MEANS CONDITION MET
;    AND 0 MEANS CONDITION NOT MET.

        DETECT,SS(11),XP,60.,.2;
        ASSIGN,XX(14) = 0.,XX(15)=0.;
        TERM;
        DETECT,SS(11),XN,-60.,.2;
        ASSIGN,XX(14)=0.,XX(15)=0.;
        TERM;
        DETECT,SS(10),XP,XX(19),.1;
        ASSIGN,XX(16) = 0.,XX(15) = 0.;
        TERM;
        DETECT,SS(10),XN,XX(18),.1;
        ASSIGN,XX(16) = 0.,XX(15) = 0.;
        TERM;
        DETECT,SS(11),XN,60.,.2;
        ASSIGN,XX(14) = 1.;
        EVENT(1);
        TERM;
        DETECT,SS(11),XP,-60.,.2;
        ASSIGN,XX(14)=1.;
        EVENT(1);
        TERM;
        DETECT,SS(10),XN,XX(19),.1;
        ASSIGN,XX(16)=1.;
        EVENT(1);
        TERM;
        DETECT,SS(10),XP,XX(18),.1;
        ASSIGN,XX(16) = 1.;
        EVENT(1);
```

```
        TERM;

; CMMCA FLIGHT PATH
;
; AZIMUTH DEVIATION CONTROL LAW

        DETECT,SS(11),XN,XX(23),.2;
        EVENT(2);
        TERM;
        DETECT,SS(11),XP,XX(22),.2;
        EVENT(2);
        TERM;
        DETECT,SS(11),XP,XX(23),.2;
        EVENT(3);
        TERM;
        DETECT,SS(11),XN,XX(22),.2;
        EVENT(3);
        TERM;

; CMMCA REACHED APPROPRIATE BANK ANGLE

        DETECT,SS(8),X,XX(20),.1;
        EVENT(4)
        TERM;

; RANGE DEVIATION CONTROL LAW

        DETECT,SS(10),XN,XX(18),.1;
        ASSIGN,ATASI=-.75,XX(21)=0.;
        TERM;
        DETECT,SS(10),XP,XX(19),.1;
        ASSIGN,ATASI=.5,XX(21)=0.;
        TERM;
        DETECT,SS(10),XP,XX(18),.1;
        EVENT(5);
        TERM;
        DETECT,SS(10),XN,XX(19),.1;
        EVENT(5);
        TERM;
        DETECT,SS(12),X,CTAS,.1;
        EVENT(6);
        TERM;

        ENDNETWORK;
FIN;
```

```
;*************************************************************
;*                                                           *
;*                                                           *
;*    DESCRIPTION:  The FORTRAN part of the SLAM II          *
;*                  simulation follows.  The only part       *
;*                  used for this thesis was the portion     *
;*                  providing x-y position for the cruise    *
;*                  missile at certain time intervals.       *
;*                                                           *
;*************************************************************

      PROGRAM MAIN
      DIMENSION NSET(10000)
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON QSET(10000)
      EQUIVALENCE(NSET(1),QSET(1))
      NNSET=10000
      NCRDR=5
      NPRNT=6
      NTAPE=7
      NPLOT=2
      OPEN(10,FILE='INPUT.DAT',STATUS='NEW')
      OPEN(11,FILE='CMMCA.OUT',STATUS='NEW')
      OPEN(12,FILE='CONE.OUT',STATUS='NEW')
C     OPEN(14,FILE='DESIGN2.DAT',STATUS='OLD')
      CALL SLAM
      STOP
      END
C
C
C
      SUBROUTINE EVENT(I)
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
       EQUIVALENCE (XX(1),ATASI),(XX(2),ATH),(XX(3),DCA),
     &             (XX(5),WSPEED),(XX(6),WDIR),(XX(7),CTAS),
     &             (SS(12),AGS),
     &             (SS(1),CEPOS),(SS(2),CNPOS),(SS(3),CTC),
     &             (SS(4),AEPOS),(SS(5),ANPOS),(SS(6),ATC),
     &             (SS(7),ATAS),(SS(8),ABANGLE),(SS(9),CBANGLE),
     &             (DD(1),CEVEL),(DD(2),CNVEL).(DD(3),CTCC),
     &             (DD(4),AEVEL),(DD(5),ANVEL),(DD(6),ATCC)
      GOTO(1,2,3,4,5,6),I

    1   IF (XX(14) .EQ. 1 .AND. XX(16) .EQ. 1) XX(15) = 1.
        RETURN

    2   XX(20)=SIGN(30.,SS(11))
        XX(17)=SIGN(XX(12),XX(20))
```

```
      RETURN

    3 IF (CBANGLE .EQ. 0) THEN
      XX(20)=0.
      XX(17)=0.
      ENDIF
      RETURN

    4 XX(17)=0.
      RETURN

    5 IF((SS(12) - CTAS).GT. .1) THEN
        ATASI=-.75
        XX(21)=1.
      ENDIF
      IF ((SS(12) - CTAS).LT.-.1) THEN
        ATASI=.5
        XX(21)=1
      ENDIF
      RETURN

    6 IF (XX(21) .EQ. 1) THEN
        ATASI=0.
        XX(21)=0.
      ENDIF
      RETURN
        END
C
C
      SUBROUTINE INTLC
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
       EQUIVALENCE (XX(1),ATASI),(XX(2),ATH),(XX(3),DCA),
     &             (XX(5),WSPEED),(XX(6),WDIR),(XX(7),CTAS),
     &             (SS(12),AGS),
     &             (SS(1),CEPOS),(SS(2),CNPOS),(SS(3),CTC),
     &             (SS(4),AEPOS),(SS(5),ANPOS),(SS(6),ATC),
     &             (SS(7),ATAS),(SS(8),ABANGLE),(SS(9),CBANGLE),
     &             (DD(1),CEVEL),(DD(2),CNVEL),(DD(3),CTCC),
     &             (DD(4),AEVEL),(DD(5),ANVEL),(DD(6),ATCC)

      REAL COSD
      EXTERNAL COSD
      REAL SIND
      EXTERNAL SIND
      PI = ACOS(-1.)

C***** READ IN DESIGN POINTS

C     READ(14,*) XX(96), XX(97), XX(98), XX(99), XX(100)
```

```
C***** SET THE FACTOR LEVELS

C     WDIR = 90*XX(96) + 180
C     WSPEED = 25*XX(97) + 50
C     XX(12) = 300*XX(98) + 600
C     XX(22) = 5*XX(99) + 10
C     XX(23) = -XX(22)
C     AEPOS = 2*XX(100)
      WDIR = 0.0
      WSPEED = 0.0
      XX(12) = 900.0
      XX(22) = 15.0
      XX(23) = -XX(22)
      AEPOS = 0.0

C***** CONVERT NAUTICAL MILES/HR TO NAUTICAL MILES/MINUTE

      WSPEED = WSPEED/60.
      CTAS = CTAS/60.

C***** SET CMMCA GROUND SPEED BASED ON INITIAL WIND & CM SPEED

        DCA = (ASIN((WSPEED/CTAS)*SIND(WDIR-ATC)))*180/PI
        ATH = ATC + DCA
        AGS = CTAS*COSD(ATH-ATC) - WSPEED*COSD(WDIR-ATC)
        ATAS = (CTAS + WSPEED*COSD(WDIR - ATC))/COSD(ATH - ATC)

      RETURN
      END
C
C
C
      SUBROUTINE STATE
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
       EQUIVALENCE (XX(1),ATAS1),(XX(2),ATH),(XX(3),DCA),
     &             (XX(5),WSPEED),(XX(6),WDIR),(XX(7),CTAS),
     &             (SS(12),AGS),
     &             (SS(1),CEPOS),(SS(2),CNPOS),(SS(3),CTC),
     &             (SS(4),AEPOS),(SS(5),ANPOS),(SS(6),ATC),
     &             (SS(7),ATAS),(SS(8),ABANGLE),(SS(9),CBANGLE),
     &             (DD(1),CEVEL),(DD(2),CNVEL),(DD(3),CTCC),
     &             (DD(4),AEVEL),(DD(5),ANVEL),(DD(6),ATCC)

       REAL SIND
       EXTERNAL SIND
       REAL COSD
       EXTERNAL COSD
       REAL TAND
       EXTERNAL TAND
```

```
      REAL ATAN2D
      EXTERNAL ATAN2D
      REAL PI,DCA,X,Y,Z
      INTEGER N,M
      PI = ACOS(-1.)


C     CRUISE MISSILE VARIABLES

C     DUE TO NO WIND ASSUMPTION FOR CRUISE MISSILE ALTITUDE
C     MISSILE TAS = MISSILE GS AND MISSILE TRUE COURSE (MTC) EQUALS
C     MISSILE TRUE HEADING.

      CEVEL = CTAS * SIND(CTC)
      CNVEL = CTAS * COSD(CTC)
      CTCC = (180. * XX(4) * TAND(CBANGLE))/(PI*CTAS)
      N = 1
      XX(11)=CTC
      IF ((CTC .GT. 720) .OR. (CTC .LT. -360)) THEN
         N = INT(ABS(CTC/360)) + 1
      ENDIF
      IF (CTC .GT. 360.) XX(11)= CTC - N * 360.
      IF (CTC .LT. 0.) XX(11)= CTC + N * 360.

C     CMMCA VARIABLES

      DD(7) = XX(1)
      DD(8) = XX(17)

C      WIND EFFECTS ON CMMCA TRUE HEADING AND TAS

      DCA = (ASIN((WSPEED/ATAS)*SIND(WDIR - ATC)))*180/PI
      ATH = ATC + DCA
      AGS = ATAS*COSD(ATH - ATC) - WSPEED*COSD(WDIR - ATC)

      AEVEL = AGS * SIND(ATC)
      ANVEL = AGS * COSD(ATC)
      ATCC = (180. * XX(4) * TAND(ABANGLE))/(PI*AGS)
      M = 1

      IF ((ATC .GT. 720) .OR. (ATC .LT. -360)) THEN
         M = IN  BS(ATC/360)) + 1
      ENDIF
      IF (ATC .GT. 360.) ATC = ATC - M * 360.
      IF (ATC .LT. 0.) ATC = ATC + M * 360.

      SS(10)=SQRT((CEPOS-AEPOS)**2 + (CNPOS-ANPOS)**2 +
     &            (28000/6076)**2 )
      SS(11)= ATAN2D((CEPOS-AEPOS),(CNPOS-ANPOS)) - ATH
      IF (SS(11) .LT. -180) SS(11)=SS(11) + 360.
```

```
C       OUTPUT DATA TO FILES CM.OUT AND CMMCA.OUT

        X = TNOW + .000004
        Y = AMOD(X,.1)
        IF ((Y .LE. .001) .AND. (TNOW .GT. Z)) THEN
           CBANGLE=CBANGLE*PI/180
           WRITE (10,100)  CBANGLE,CTAS,CEPOS,CNPOS
           Z = TNOW
        ENDIF
100     FORMAT (3(5X,F9.4))
        RETURN
        END

       SUBROUTINE OTPUT
       COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
      1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
        EQUIVALENCE (XX(1),ATASI),(XX(2),ATH),(XX(3),DCA),
      &             (XX(5),WSPEED),(XX(6),WDIR),(XX(7),CTAS),
      &             (SS(12),AGS),
      &             (SS(1),CEPOS),(SS(2),CNPOS),(SS(3),CTC),
      &             (SS(4),AEPOS),(SS(5),ANPOS),(SS(6),ATC),
      &             (SS(7),ATAS),(SS(8),ABANGLE),(SS(9),CBANGLE),
      &             (DD(1),CEVEL),(DD(2),CNVEL),(DD(3),CTCC),
      &             (DD(4),AEVEL),(DD(5),ANVEL),(DD(6),ATCC)

       WRITE(12,110) TTAVG(1)
110    FORMAT (4X,F9.4)
       RETURN
       END


        REAL FUNCTION SIND(X)
        REAL X, PI
        INTRINSIC SIN
        PI = ACOS(-1.)

        SIND = SIN(X*PI/180)
        END

        REAL FUNCTION COSD(X)
        REAL X, PI
        INTRINSIC COS
        PI = ACOS(-1.)

        COSD = COS(X*PI/180)
        END

        REAL FUNCTION TAND(X)
        REAL X, PI
```

```
INTRINSIC TAN
PI = ACOS(-1.)

TAND = TAN(X*PI/180)
END

REAL FUNCTION ATAN2D(Y,X)
REAL X, PI, Y
INTRINSIC ATAN2
PI = ACOS(-1.)

ATAN2D = ATAN2(Y,X)*180/PI
END
```

Appendix C. Curve Fitting

These are the calculations for deriving the optimization algorithm's three point parabola curve fit. Eqs (C.1), (C.2) and (C.3) define the three points that will fit the curve. The first point will be where the step size is zero, in other words the maneuver vector with no change. The objective functional value will be computed for this step size of zero using Eqs (3.29) and (3.7). The next two points represented by lambda one and lambda two will use step sizes of different value. Again each of these step sizes will be used to compute a new objective functional value.

$$g_0 = J(\mathbf{x}(0)) \tag{C.1}$$

$$g_1 = J(\mathbf{x}(\lambda_1)) \tag{C.2}$$

$$g_2 = J(\mathbf{x}(\lambda_2)) \tag{C.3}$$

The equation for a parabola follows with the initial condition that g_0 equals c at lambda equals zero.

$$g = a\lambda^2 + b\lambda + c \qquad g_o = c \tag{C.4}$$

It follows from algebra that,

$$g_1 - g_o = a\lambda_1^2 + b\lambda_1 \tag{C.5}$$

and

$$g_2 - g_o = a\lambda_2^2 + b\lambda_2 \tag{C.6}$$

Given

$$f_i = \frac{g_i - g_0}{\lambda_i} \tag{C.7}$$

it follows that

$$\begin{bmatrix} \lambda_1 & 1 \\ \lambda_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \qquad \text{(C.8)}$$

Using matrix multiplication yields

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\lambda_1 - \lambda_2} \begin{bmatrix} 1 & -1 \\ -\lambda_2 & \lambda_1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \qquad \text{(C.9)}$$

To find the optimal lambda from the fitted parabola requires taking the first derivative of Eq (C.4).

$$g'(\lambda) = 2a\lambda + b = 0 \quad \Rightarrow \lambda_{opt} = \frac{-b}{2a} \qquad \text{(C.10)}$$

After a few algebraic manipulations a final form is derived that can be used in the optimization algorithm.

$$\lambda_{opt} = -\frac{1}{2}\left(\frac{b}{a}\right) = -\frac{1}{2}\begin{bmatrix} -\lambda_2 f_1 + \lambda_1 f_2 \\ f_1 - f_2 \end{bmatrix} \qquad \text{(C.11)}$$

$$= -\frac{1}{2}\,\frac{\frac{\lambda_1}{\lambda_2}(g_2-g_0) - \frac{\lambda_2}{\lambda_1}(g_1-g_0)}{\frac{(g_1-g_0)}{\lambda_1} - \frac{(g_2-g_0)}{\lambda_2}} \qquad \text{(C.12)}$$

$$= \frac{1}{2}\,\frac{\lambda_1^2(g_2-g_0) + \lambda_2^2(g_1-g_0)}{\lambda_2(g_1-g_0) - \lambda_1(g_2-g_0)} \qquad \text{(C.13)}$$

This is the derivation of the modified parabola fit using the gradient or slope, JSTOP and an additional point λ_1. The slope is derived at the first point λ_0 which is equal to zero.

$$g(\lambda) = a\lambda^2 + b\lambda + g_0 \qquad \text{(C.14)}$$

$$g'(\lambda) = 2a\lambda + b$$

$$g'(0) = b = g_0' = -JSTOP$$

$$\Rightarrow g(\lambda) = a\lambda^2 + g_0'\lambda + g_0 \qquad \text{(C.15)}$$

$$g(\lambda_1) = a\lambda_1^2 + g_0'\lambda_1 + g_0 = g_1$$

$$\Rightarrow a = \frac{g_1 - g_0 - g_0' \lambda_1}{\lambda_1^2} \qquad \text{(C.16)}$$

$$\lambda_{opt} - g'(\lambda) - 2a\lambda + g_0' - 0$$

$$\Rightarrow \lambda_{opt} = \frac{-g_0'}{2a} = \frac{-g_0' \lambda_1^2}{2(g_1 - g_0 - g_0' \lambda_1)} \qquad \text{(C.17)}$$

$$= \frac{g_0' \lambda_1^2}{2(g_0' \lambda_1 + g_0 - g_1)}$$

$$- \frac{\lambda_1}{2} + \frac{(g_1 - g_0)\lambda_1}{2(g_0' \lambda_1 + g_0 - g_1)}$$

$$= \frac{1}{2}\left[\lambda_1 + \frac{g_1 - g_0}{g_0' + \left(\frac{g_0 - g_1}{\lambda_1}\right)}\right] \qquad \text{(C.18)}$$

Appendix D. Turn Results

The results for six test runs are presented here. Tables present five columns of information. The first column depicts the time. The second and third columns describe the bank (in degrees) and the speed (in knots TAS) the CMMCA should use to follow the cruise missile. Negative banks are left turns and positive banks right turns. The last two columns, range (in nm's) and azimuth (in degrees) are measured from the CMMCA to the CM. Azimuth is an angle measured from the nose of the CMMCA to the position of the CM. Negative azimuths represent angles to the left while positive values represent angles to the right.

In the first four test runs the cruise missile is flying straight and level at 400 knots TAS. The first table displays the results when the CMMCA starts to the left of the CM, 9.5 miles slant range behind at 400 knots TAS. The second table displays the results when the CMMCA is positioned to the right at the same distance and speed. The third table depicts the result of the CMMCA to the right and in front of the CM. In the fourth table the CMMCA is started to the left and at a slower speed of 360 knots TAS.

The fifth table depicts the result for a 90 degree turn performed by the cruise missile. The last table shows results for a 270 degree turn.

JSTOP= 4.514542
# ITERATIONS =750

TIME	BANK	SPEED	RANGE	THETA
.1	32.4	240.5	9.5	22.8
.2	9.1	351.6	9.6	11.4
.3	4.1	434.0	9.6	8.1
.4	1.9	450.0	9.4	5.9
.5	0.7	453.1	9.3	4.2
.6	0.0	453.1	9.1	2.6
.7	-0.5	451.6	9.0	1.1
.8	-0.8	449.6	8.8	-0.3
.9	-1.0	447.2	8.7	-1.7
1.0	-1.2	444.6	8.6	-3.1
1.1	-1.4	442.1	8.5	-4.4
1.2	-1.6	439.5	8.4	-5.7
1.3	-1.8	437.1	8.3	-6.9
1.4	-2.1	434.8	8.3	-8.1
1.5	-2.4	432.6	8.2	-9.1
1.6	-2.7	430.5	8.2	-10.0
1.7	-3.0	428.5	8.1	-10.7
1.8	-3.3	426.6	8.1	-11.4
1.9	-3.6	424.7	8.0	-11.8
2.0	-3.9	422.8	8.0	-12.1
2.1	-4.1	421.0	8.0	-12.3
2.2	-4.2	419.2	8.0	-12.3
2.3	-4.3	417.3	8.0	-12.2
2.4	-4.4	415.5	8.0	-11.9
2.5	-4.3	413.7	7.9	-11.5
2.6	-4.3	411.9	7.9	-11.1
2.7	-4.1	410.2	7.9	-10.5
2.8	-4.0	408.5	7.9	-9.9
2.9	-3.7	406.9	7.9	-9.3
3.0	-3.5	405.4	7.9	-8.6
3.1	-3.2	404.0	7.9	-7.9
3.2	-2.9	402.7	7.9	-7.2
3.3	-2.6	401.5	7.9	-6.5
3.4	-2.3	400.5	7.9	-5.8
3.5	-2.0	399.6	7.9	-5.2
3.6	-1.7	398.8	7.9	-4.6
3.7	-1.5	398.2	7.9	-4.0
3.8	-1.2	397.6	7.9	-3.5
3.9	-1.0	397.2	7.9	-3.0
4.0	-0.7	396.9	7.9	-2.6
4.1	-0.5	396.7	7.9	-2.2
4.2	-0.4	396.5	7.9	-1.8
4.3	-0.2	396.5	7.9	-1.5
4.4	-0.1	396.5	7.9	-1.2

4.5	0.0	396.5	7.9	-0.9
4.6	0.1	396.7	7.9	-0.7
4.7	0.2	396.8	7.9	-0.5
4.8	0.2	397.0	7.9	-0.3
4.9	0.3	397.2	7.9	-0.1
5.0	0.3	397.4	7.9	0.0
5.1	0.3	397.6	8.0	0.1
5.2	0.3	397.9	8.0	0.3
5.3	0.3	398.1	8.0	0.4
5.4	0.3	398.4	8.0	0.5
5.5	0.2	398.7	8.0	0.6
5.6	0.2	398.9	8.0	0.7
5.7	0.2	399.2	8.0	0.9
5.8	0.1	399.5	8.0	1.0
5.9	0.1	399.7	8.0	1.1
6.0	0.0	399.9	8.0	1.3

JSTOP= 0.2893718
# ITERATIONS =750

TIME	BANK	SPEED	RANGE	THETA
.1	-31.1	503.2	9.3	-28.2
.2	-15.7	484.5	9.2	-22.4
.3	-9.3	473.3	9.0	-18.9
.4	-5.5	463.8	8.8	-16.4
.5	-3.0	455.4	8.6	-14.4
.6	-1.4	447.4	8.5	-12.8
.7	-0.3	439.2	8.4	-11.5
.8	0.4	428.0	8.2	-10.4
.9	0.9	411.0	8.2	-9.3
1.0	1.3	394.5	8.1	-8.4
1.1	1.5	384.6	8.1	-7.5
1.2	1.7	379.7	8.0	-6.8
1.3	1.8	377.7	8.0	-6.1
1.4	1.9	377.2	8.0	-5.5
1.5	1.9	377.7	8.0	-4.9
1.6	1.9	378.7	8.1	-4.5
1.7	1.9	380.1	8.1	-4.0
1.8	1.9	381.8	8.1	-3.6
1.9	1.9	383.7	8.1	-3.2
2.0	1.8	385.6	8.1	-2.9
2.1	1.8	387.6	8.1	-2.6
2.2	1.7	389.5	8.1	-2.3
2.3	1.7	391.3	8.1	-2.0
2.4	1.6	393.0	8.1	-1.8
2.5	1.6	394.5	8.1	-1.5
2.6	1.5	395.9	8.1	-1.4
2.7	1.4	397.1	8.1	-1.2
2.8	1.4	398.1	8.1	-1.0
2.9	1.3	399.0	8.1	-0.9
3.0	1.3	399.7	8.1	-0.7
3.1	1.2	400.3	8.1	-0.6
3.2	1.2	400.8	8.1	-0.5
3.3	1.1	401.2	8.1	-0.4
3.4	1.1	401.5	8.1	-0.3
3.5	1.0	401.7	8.1	-0.2
3.6	1.0	401.8	8.1	-0.2
3.7	0.9	401.9	8.1	-0.1
3.8	0.9	402.0	8.1	-0.1
3.9	0.8	402.0	8.0	0.0
4.0	0.8	402.0	8.0	0.0
4.1	0.7	402.0	8.0	0.0
4.2	0.7	401.9	8.0	0.1
4.3	0.7	401.8	8.0	0.1
4.4	0.6	401.8	8.0	0.1

4.5	0.6	401.7	8.0	0.1
4.6	0.5	401.6	8.0	0.1
4.7	0.5	401.5	8.0	0.1
4.8	0.5	401.4	8.0	0.1
4.9	0.4	401.3	8.0	0.2
5.0	0.4	401.2	8.0	0.2
5.1	0.3	401.0	8.0	0.2
5.2	0.3	400.9	8.0	0.2
5.3	0.3	400.8	8.0	0.2
5.4	0.2	400.7	8.0	0.2
5.5	0.2	400.6	8.0	0.3
5.6	0.2	400.5	8.0	0.3
5.7	0.1	400.4	8.0	0.3
5.8	0.1	400.3	8.0	0.4
5.9	0.0	400.1	8.0	0.5
6.0	0.0	400.0	8.0	0.5

JSTOP= 80.99321
# ITERATIONS =750

TIME	BANK	SPEED	RANGE	THETA
.1	-11.7	343.1	8.6	-142.4
.2	-9.9	351.0	8.5	-138.6
.3	-6.9	360.5	8.4	-136.0
.4	-3.1	370.1	8.2	-134.7
.5	0.9	378.9	8.1	-134.8
.6	4.8	386.1	8.0	-136.1
.7	8.2	391.3	8.0	-138.4
.8	11.0	394.5	7.9	-141.6
.9	12.8	395.8	7.9	-145.3
1.0	13.6	395.3	7.9	-149.1
1.1	13.1	393.5	7.9	-152.6
1.2	11.6	390.8	8.0	-155.6
1.3	9.3	387.6	8.0	-157.8
1.4	6.3	384.5	8.1	-159.0
1.5	3.1	381.7	8.1	-159.2
1.6	-0.1	379.5	8.2	-158.3
1.7	-3.1	378.0	8.3	-156.5
1.8	-5.6	377.4	8.3	-154.0
1.9	-7.6	377.6	8.4	-150.9
2.0	-8.8	378.5	8.5	-147.5
2.1	-9.4	379.9	8.5	-144.0
2.2	-9.2	381.9	8.5	-140.6
2.3	-8.5	384.0	8.6	-137.6
2.4	-7.3	386.3	8.6	-135.0
2.5	-5.9	388.5	8.5	-133.0
2.6	-4.4	390.5	8.5	-131.5
2.7	-2.9	392.3	8.5	-130.5
2.8	-1.5	394.0	8.4	-130.1
2.9	-0.2	395.4	8.4	-130.0
3.0	0.7	396.6	8.3	-130.3
3.1	1.5	397.7	8.3	-130.8
3.2	2.0	398.7	8.3	-131.5
3.3	2.2	399.6	8.2	-132.3
3.4	2.3	400.4	8.2	-133.1
3.5	2.2	401.3	8.2	-133.9
3.6	2.0	402.2	8.2	-134.6
3.7	1.7	403.1	8.2	-135.2
3.8	1.3	404.1	8.2	-135.7
3.9	0.9	405.1	8.2	-136.1
4.0	0.5	406.2	8.2	-136.3
4.1	0.1	407.4	8.2	-136.4
4.2	-0.4	408.7	8.2	-136.4
4.3	-0.8	409.9	8.2	-136.3
4.4	-1.3	411.3	8.2	-136.1

4.5	-1.9	412.6	8.2	-135.8
4.6	-2.4	413.9	8.2	-135.4
4.7	-3.1	415.2	8.3	-134.8
4.8	-3.9	416.4	8.2	-134.1
4.9	-4.9	417.4	8.2	-133.3
5.0	-5.9	418.2	8.2	-132.2
5.1	-7.2	418.7	8.2	-131.0
5.2	-8.6	418.8	8.1	-129.4
5.3	-10.1	418.3	8.1	-127.7
5.4	-11.5	417.2	8.0	-125.6
5.5	-12.8	415.4	7.8	-123.4
5.6	-13.4	413.0	7.7	-121.0
5.7	-12.9	410.0	7.5	-118.9
5.8	-10.7	406.7	7.2	-117.3
5.9	-6.3	403.4	7.0	-116.8
6.0	-2.9	400.9	6.7	-117.7

JSTOP= 2.400061
# ITERATIONS =750

TIME	BANK	SPEED	RANGE	THETA
.1	32.8	239.3	9.6	22.5
.2	8.7	380.8	9.6	11.1
.3	4.1	439.4	9.5	8.0
.4	1.9	450.2	9.4	5.9
.5	0.6	452.5	9.2	4.1
.6	-0.1	452.1	9.1	2.5
.7	-0.6	450.6	9.0	1.1
.8	-1.0	448.4	8.8	-0.3
.9	-1.2	446.0	8.7	-1.7
1.0	-1.4	443.5	8.6	-3.0
1.1	-1.7	441.0	8.5	-4.2
1.2	-1.9	438.5	8.4	-5.4
1.3	-2.1	436.1	8.3	-6.5
1.4	-2.4	433.7	8.3	-7.5
1.5	-2.7	431.5	8.2	-8.4
1.6	-3.0	429.3	8.1	-9.1
1.7	-3.3	427.3	8.1	-9.8
1.8	-3.5	425.2	8.1	-10.2
1.9	-3.7	423.2	8.0	-10.6
2.0	-3.9	421.3	8.0	-10.8
2.1	-4.0	419.4	8.0	-10.9
2.2	-4.1	417.5	8.0	-10.8
2.3	-4.1	415.7	8.0	-10.7
2.4	-4.1	414.0	7.9	-10.4
2.5	-4.0	412.3	7.9	-10.1
2.6	-3.9	410.8	7.9	-9.7
2.7	-3.7	409.3	7.9	-9.2
2.8	-3.5	407.9	7.9	-8.7
2.9	-3.3	406.6	7.9	-8.2
3.0	-3.0	405.5	7.9	-7.6
3.1	-2.8	404.4	7.9	-7.0
3.2	-2.5	403.4	7.9	-6.5
3.3	-2.2	402.6	7.9	-6.0
3.4	-2.0	401.8	7.9	-5.4
3.5	-1.7	401.1	7.9	-4.9
3.6	-1.5	400.4	7.9	-4.5
3.7	-1.3	399.8	7.9	-4.0
3.8	-1.0	399.3	7.9	-3.6
3.9	-0.9	398.7	7.9	-3.2
4.0	-0.7	398.2	7.9	-2.9
4.1	-0.5	397.6	7.9	-2.5
4.2	-0.4	397.0	7.9	-2.3
4.3	-0.3	396.4	7.9	-2.0
4.4	-0.2	395.8	7.9	-1.7

4.5	-0.1	395.0	7.9	-1.5
4.6	0.0	394.2	7.9	-1.3
4.7	0.0	393.3	7.9	-1.1
4.8	0.1	392.3	7.9	-1.0
4.9	0.1	391.2	7.9	-0.8
5.0	0.1	390.0	8.0	-0.7
5.1	0.2	388.6	8.0	-0.6
5.2	0.2	387.2	8.0	-0.4
5.3	0.2	385.6	8.0	-0.3
5.4	0.2	383.8	8.0	-0.2
5.5	0.1	381.9	8.1	-0.1
5.6	0.1	379.9	8.1	0.0
5.7	0.1	377.8	8.1	0.1
5.8	0.1	375.5	8.2	0.3
5.9	0.0	373.2	8.2	0.4
6.0	0.0	367.4	8.3	0.5

JSTOP= 9.1502592E-02
# ITERATIONS =429

TIME	BANK	SPEED	RANGE	THETA
.1	-3.7	424.2	8.0	0.5
.2	-2.9	417.6	7.9	1.5
.3	-2.3	409.6	7.9	2.4
.4	-1.8	401.7	7.9	3.2
.5	-1.3	395.3	7.9	3.8
.6	-0.8	390.6	7.9	4.4
.7	-0.4	387.0	8.0	4.9
.8	0.0	384.1	8.0	5.2
.9	0.4	381.4	8.0	5.4
1.0	0.8	378.7	8.0	5.5
1.1	1.3	374.9	8.1	5.7
1.2	1.9	372.5	8.1	6.1
1.3	2.4	370.1	8.2	6.8
1.4	3.1	367.8	8.2	7.7
1.5	3.8	365.5	8.2	8.8
1.6	4.6	363.4	8.2	9.9
1.7	5.5	361.5	8.2	11.1
1.8	6.5	359.8	8.2	12.3
1.9	7.6	358.3	8.2	13.4
2.0	8.7	357.0	8.2	14.4
2.1	9.8	356.0	8.2	15.3
2.2	10.8	355.4	8.1	16.0
2.3	11.8	355.1	8.1	16.6
2.4	12.5	355.4	8.0	17.0
2.5	12.8	356.2	7.9	17.5
2.6	12.7	357.8	7.8	17.9
2.7	12.1	359.9	7.8	18.3
[illegible]	11.2	362.7	7.8	18.8
2.9	9.8	366.1	7.8	19.4
3.0	8.2	370.2	7.8	20.3
3.1	6.3	376.1	7.8	21.6
3.2	4.3	384.4	7.9	23.3
3.3	2.2	393.5	7.9	25.7
3.4	1.1	398.8	8.0	28.4

JSTOP=1.01152
# ITERATIONS =500

TIME	BANK	SPEED	RANGE	THETA
.1	12.3	390.6	8.0	-1.9
.2	10.0	396.0	8.0	-5.5
.3	8.2	400.1	8.0	-8.6
.4	6.9	403.3	8.0	-11.5
.5	5.9	405.7	8.0	-14.2
.6	5.3	407.6	8.1	-16.9
.7	5.0	408.9	8.1	-19.5
.8	4.9	410.0	8.1	-22.1
.9	5.0	410.7	8.1	-24.9
1.0	5.2	411.2	8.2	-27.8
1.1	9.3	390.1	8.3	-31.1
1.2	10.2	392.5	8.3	-34.9
1.3	11.4	394.3	8.4	-38.8
1.4	12.9	395.8	8.5	-42.8
1.5	14.5	397.1	8.5	-47.1
1.6	16.1	398.3	8.6	-51.8
1.7	17.6	399.6	8.6	-56.9
1.8	18.9	400.9	8.6	-62.4
1.9	19.9	402.4	8.6	-68.2
2.0	20.4	404.0	8.6	-74.3
2.1	20.5	405.9	8.7	-80.4
2.2	20.0	407.9	8.7	-86.4
2.3	19.0	410.0	8.7	-92.2
2.4	17.5	412.3	8.7	-97.6
2.5	15.7	414.7	8.7	-102.4
2.6	13.5	417.2	8.7	-106.7
2.7	11.1	419.6	8.7	-110.4
2.8	8.4	422.0	8.6	-113.5
2.9	5.6	424.2	8.5	-116.1
3.0	2.6	426.0	8.4	-118.1
3.1	-0.5	427.5	8.3	-119.6
3.2	-3.7	428.5	8.1	-120.8
3.3	-6.7	428.8	7.8	-121.7
3.4	-9.2	428.4	7.6	-122.6
3.5	-11.0	427.2	7.3	-123.8
3.6	-11.3	425.4	7.0	-125.7
3.7	-9.4	422.9	6.8	-128.9
3.8	-4.6	419.7	6.7	-134.0
3.9	4.0	415.9	6.7	-141.6
4.0	21.0	410.5	6.9	-152.9
4.1	77.2	399.0	7.2	162.6
4.2	39.0	395.4	7.5	117.9
4.3	35.3	391.3	7.7	104.9
4.4	31.9	389.0	7.8	93.5

4.5	28.5	388.0	7.8	83.5
4.6	25.5	387.9	7.8	74.8
4.7	23.1	388.1	7.8	67.0
4.8	21.1	388.5	7.8	59.9
4.9	19.4	389.1	7.8	53.4
5.0	17.8	389.7	7.8	47.4
5.1	16.4	390.5	7.8	42.0
5.2	15.0	391.3	7.8	37.1
5.3	13.6	392.3	7.8	32.7
5.4	12.1	393.3	7.8	29.0
5.5	10.7	394.4	7.8	25.9
5.6	5.8	396.4	7.8	23.9
5.7	4.4	397.3	7.9	22.8
5.8	3.0	398.2	7.9	22.0
5.9	1.5	399.1	7.9	21.5
6.0	0.6	399.8	7.9	21.3

Bibliography

1. Baker, W. P. CMMCA Cost Functional. Unpublished report. Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1988.

2. Bekey, George A. "Simulability of Dynamical Systems," Transactions of The Society for Computer Simulation, 2: 57-70 (May 1985).

3. Heavner, Capt Leonard G. An Investigation into the Application of Dynamic Programming to Determine Optimal Controls for Tracking a Cruise Missile Maneuver by CMMCA. MS thesis, AFIT/GST/ENS/89M-6. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1989.

4. Hillier, Frederick S. and Lieberman, Gerald J. Introduction to Operations Research (Third Edition). Oakland: Holden-Day, Inc., 1980.

5. Horton, Capt Robert W., ARIA Programs Division. Personal Interview. 4950th Test Wing, Wright-Patterson AFB OH, 7 July 1989.

6. Kirk, Donald E. Optimal Control Theory. Englewood Cliffs: Prentice-Hall, Inc. 1970.

7. Kunz, Kaiser S. Numerical Analysis. New York: McGraw-Hill Book Company, Inc., 1957.

8. Oyola, Capt Renee, Navigator 4952nd Test Squadron. Personal Interview. 4950th Test Wing, Wright-Patterson AFB OH, 4 August 1989.

9. Pritsker, Alan B. Introduction to Simulation and SLAM II (Third Edition). New York: Halsted Press, 1986.

10. Sargent,Robert G. "A tutorial on validation and verification of simulation models," Proceedings of the 1988 Winter Simulation Conference. 33-39.

11. Schuppe, Thomas F. Analysis of CMMCA Tracking System. Unpublished report. Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1988.

12. Sheppard Sallie "Applying software engineering to simulation," Simulation, 40: 13-19 (January 1983).

13. Stewart, Stephen J. "Flight simulation in museums," Simulation, 51: 235-241 (December 1988).

Vita

Captain Antoine M. Garton was born on 22 December 1959 in St. Denis, France. He graduated from High School in Wuerzburg, Germany, in 1977 and attended the United States Air Force Academy, from which he received the degree of Bachelor of Science in General Engineering in June 1982. After graduation, he attended Undergraduate Navigator Training at Sacramento, California, and received his wings in January 1983. He then served as a KC-135 navigator, instructor, and evaluator in the 43rd Air Refueling Squadron, Fairchild AFB, WA, until entering the School of Engineering, Air Force Institute of Technology, in August 1988.

Permanent address: 8273 Twin Pointe Circle
Indianapolis, IN 46236

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GST/ENS/90M-6	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (*If applicable*): AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (*City, State, and ZIP Code*): Air Force Institute of Technology, Wright-Patterson AFB, OH 45433	7b. ADDRESS (*City, State, and ZIP Code*)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*): Using Simulation to Determine a Strategy for Positively Tracking a Cruise Missile by CMMCA (U)

12. PERSONAL AUTHOR(S): Garton, Antoine M., Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Masters Thesis	FROM _____ TO _____	1990 March	114

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	
17	07	03	Radar Tracking, Guidance, Cruise Missiles, Control Theory, Algorithms, Computerized Simulation
12	04		

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

see reverse

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (*Include Area Code*)	22c. OFFICE SYMBOL
Thomas F. Schuppe, Lt Col, USAF	513-255-3362	AFIT/ENS

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED